FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Programação para Jovens: Conteúdos, Atividades, Estratégias e Ferramentas

Nuno Filipe Gomes dos Santos

DISSERTAÇÃO

Mestrado Integrado em Engenharia Informática

Orientador: Ademar Aguiar

13 de Fevereiro de 2017

Programação para Jovens: Conteúdos, Atividades, Estratégias e Ferramentas

Nuno Filipe Gomes dos Santos

Mestrado Integrado em Engenharia Informática

Aprovado em provas públicas pelo Júri:

Presidente: João Tiago Pinheiro Neto Jacob

Arguente: Luís Borges Gouveia

13 de Fevereiro de 2017

Resumo

A programação de computadores é uma competência com uma crescente importância na formação de uma pessoa, em particular nos jovens. Mais do que habilitar para uma melhor utilização da enorme capacidade e funcionalidades computacionais hoje disponíveis, permite por si só desenvolver o raciocínio lógico agregado à capacidade de resolução de problemas.

Como tal, o ensino da programação tem vindo a ser realizado a jovens e crianças cada vez mais novas, de todo o globo, com várias iniciativas como o code.org, Kodu Game Lab, Hour of Code, Khan Academy e novas ferramentas para o ensino da programação como o Scratch, Alice ou ToonTalk. Estas ferramentas e tecnologias são usadas em contexto curricular e extracurricular, na escola e em casa, com as devidas adaptações em termos de conteúdos, atividades, estratégias de ensino/aprendizagem e ferramentas de programação.

Alguns países têm-se claramente destacado neste tema, tanto por avanços no ensino da tecnologia como por implementação, no seu plano curricular oficial, do ensino da programação. O Reino Unido, por exemplo, implementou o programa de ensino de programação no ano letivo de 2014. A realidade em Portugal embora semelhante, existem ainda necessidades específicas. A oferta educativa consiste em Tecnologia de Informação e Computação (ensino básico) e Aplicações Informáticas B (ensino secundário), cujos currículos podem ser melhorados, e a existência destas disciplinas tipicamente depende da oferta educativa de cada escola, motivando, ou não, os alunos que ambicionam seguir áreas tecnológicas no ensino superior.

Neste contexto, nacional e internacional, pretendeu-se começar por desenvolver um conjunto de unidades didáticas (conteúdos e atividades) para alunos e docentes, assim como um conjunto de estratégias, metas, objetivos educativos e pedagógicos para o ensino da programação a jovens. Estudos da Universidade de Minnesota referem que o tempo voa quando os alunos se divertem, num estudo que relaciona o uso de tecnologia com a absorção cognitiva, como tal moldou-se a construção deste trabalho tendo em atenção a preferência dos alunos. De modo a evitar condicionantes extra à aprendizagem por parte dos alunos, contribui-se com conteúdos em língua portuguesa, são ainda selecionadas, configuradas e integradas um conjunto de ferramentas de programação, recorrendo ao estado da arte da área. Esta dissertação visa o desenvolvimento de conteúdos e atividades, estratégias e ferramentas, para o ensino da programação a jovens entre os 12 e 18 anos de idade, em língua portuguesa.

Palavras-chave: Edutech, K-12, Programação, Jovens, Ferramentas, Linguagens

Abstract

Computer programming is an increasingly important competence in the training of a person, particularly young people. More than enabling for a better use of the enormous capacity and computational functionalities available today, it allows itself to develop the logical reasoning added to the capacity of problem solving.

As such, programming education has been delivered to increasingly young people and young people from all over the globe, with various initiatives such as code.org, Kodu Game Lab, Hour of Code, Khan Academy and new tools for Teaching programming like Scratch, Alice or ToonTalk. These tools and technologies are used in curricular and extracurricular contexts, at school and at home, with appropriate adaptations in terms of content, activities, teaching / learning strategies and programming tools.

Some countries have clearly distinguished themselves in this theme, both by advances in the teaching of technology and by the implementation, in their official curricular plan, of programming teaching. The UK, for example, has implemented the programming teaching program in the 2014 school year. The reality in Portugal although similar, still has specific needs. The educational offer consists of Information Technology and Computing (basic education) and Computer Applications B (secondary education), whose curricula can be improved, and the existence of these disciplines typically differs according to the educational offer of each school, motivating or not, the students who ambition to follow technological areas in higher education.

In this context, national and international, it was intended to begin by developing a set of didactic units (contents and activities) for students and teachers, as well as a set of strategies, goals, educational and pedagogical objectives for the teaching of programming to children and young people . Studies of the University of Minnesota report that time flies when students have fun in a study that relates the use of technology to cognitive absorption, as such the construction of this work was shaped by the preference of students. In order to avoid extra conditioning to students' learning, it contributes with content in Portuguese, Are still selected, configured and integrated a set of programming tools, using the state of the art of the area. This dissertation aims at the development of contents and activities, strategies and tools, for the teaching of programming to young people between 12 and 18 years of age, in Portuguese language.

Keywords: Edutech, K-12, Programming, Coding, Kids, Tools, Languages

ACM Computing Classification System: CCS - Social and professional topics - Professional topics - Computing education - K-12 education

Agradecimentos

Ao professor Ademar Aguiar que me deu a confiança e apoio. Acredito que são poucos os alunos que têm a oportunidade de trabalhar as suas próprias ideias. Muito obrigado!

Um viva à minha esposa pelo fantástico apoio durante todos estes anos de faculdade tu foste o MVP, aos meus pais por toda a ajuda dada e por garantirem que nada me faltasse, à minha irmã porque sem ela era diferente, à minha avó que me educou para ser uma pessoa melhor, ao meu avô que foi responsável pelo meu modo engenhoso de pensar e pela curiosidade ilimitada, ao Brian pela alegria diária em me ver.

Isto é para todos vós.

Nuno Santos

"The scientific man does not aim at an immediate result.
He does not expect that his advanced ideas will be readily taken up.
His work is like that of the planter - for the future.
His duty is to lay the foundation for those who are to come, and point the way.
He lives and labors and hopes."

Nikola Tesla

Conteúdo

Lista de Figuras

Lista de Tabelas

Abreviaturas e Símbolos

RPi	Raspberry Pi
Edutech	Tecnologia Educativa
TIC	Tecnologia da Informação e Comunicação
B-Learning	Blended Learning
E-Learning	Eletronic Learning
MOOC	Massive Open Online Course
API	Application Programming Interface
K-12	Até ao 12º ano de escolaridade
CSI Labs	Coding for Social Impact Labs
PBL	Project Based Learning
DIY	Do It Yourself

Capítulo 1

Introdução

Esta dissertação foi realizada no âmbito da unidade curricular da Dissertação, com o objetivo global de sensibilizar, promover e facilitar o ensino da programação. Em particular, um dos objetivos principais consiste na criação e disponibilização, de forma livre e aberta, de unidades didáticas, conteúdos, atividades e estratégias com o auxílio de várias ferramentas, para jovens de idades entre os 12 e os 18 anos, respetivos professores e escolas.

> "Acho que mais crianças devem aprender a programar. Eu não diria que devemos impor isso a todos. O fato de entender como funciona um computador dá-lhe um sentido real do que ele pode fazer, o que ele não pode fazer. ... Não temos programação suficiente no currículo. Mas eu certamente não coloca-lo-ía como matemática ou literacia, mas há muito material inovador que torna-o mais interessante."
>
> — Bill Gates, More kids should learn to program, CNN[1]

1.1 Contexto

A programação de computadores é uma competência com uma crescente importância na formação de uma pessoa, em particular nos jovens. Mais do que habilitar para uma melhor utilização da enorme capacidade e funcionalidades computacionais hoje disponíveis, permite por si só desenvolver o raciocínio lógico agregado à capacidade de resolução de problemas. Como tal, o ensino da programação tem vindo a ser realizado a jovens e crianças cada vez mais novas, de todo o globo, com várias iniciativas [2] [3]. Várias ferramentas e tecnologias são usadas em contexto curricular e extra-curricular, na escola e em casa, com as devidas adaptações em termos de conteúdos, atividades, estratégias de ensino/aprendizagem e ferramentas de programação.

A realidade em Portugal embora semelhante, possuem necessidades específicas. A oferta educativa consiste em Tecnologia de Informação e Computação (ensino básico [4]) e Aplicações

[1] https://www.youtube.com/watch?v=EwxSPm9rNSQ
[2] https://hourofcode.com/pt
[3] https://code.org/
[4] http://www.dge.mec.pt/matriz-curricular-do-3o-ciclo

Informáticas B (ensino secundário [5]), cujos currículos podem ser melhorados, e a existência destas disciplinas tipicamente difere de acordo com a oferta educativa de cada escola, motivando, ou não, os alunos que ambicionam seguir áreas tecnológicas no ensino superior. Existem evidências que correlacionam a má experiência com estas disciplinas com o desinteresse generalizado em atividades de programação [GPC14]. A escassez de iniciativas e ferramentas para o ensino de programação em português é também uma condicionante para o ensino de programação aos jovens. Estudos da Universidade de Minnesota [AK00] referem que o tempo voa quando os alunos se divertem, num estudo que relaciona o uso de tecnologia com a absorção cognitiva, como tal moldou-se a construção deste trabalho tendo em atenção a preferência dos alunos.

1.2 Motivação e Objetivos

No ensino superior, durante a *Talk a Bit 2015* e após uma palestra relacionada com o ensino da tecnologia no ensino básico e secundário, a oradora Fernanda Ledesma, Presidente da Direção da Associação Nacional de Professores de Informática, referiu que, e parafraseando, "a área de Ciências e Tecnologias já não possuiu a componente tecnológica". Tal afirmação, motivou à participação numa iniciativa dedicada à programação no âmbito da Universidade Júnior, permitindo aferir a relação dos jovens com a programação. O *feedback* recebido reforçou a ideia inicial, revelando, ainda, que mesmo entre estudantes do mesmo ano de escolaridade havia uma grande variabilidade de contacto com tecnologias em função da escola de que provinham: alguns tiveram TIC, outros não, e, nos casos em que tiveram a disciplina, os conteúdos abordados eram completamente díspares de uns alunos para os outros.

Neste contexto, foi identificada a necessidade de criar novas metodologias, conteúdos e até mesmo programas, destinadas a jovens e crianças entre os 12 e os 18 anos de idade e respetivos educadores, professores e encarregados de educação, com conteúdos em língua portuguesa, para que o ensino da programação seja mais apelativo e intuitivo, tudo isto aliado a um leque relevante de ferramentas para a faixa etária em questão. Assim, esta tema foi auto-proposto ao professor Ademar Aguiar que, como instigador desta problemática, aconselhou e entusiasmou ainda mais para o desenvolvimento deste tema.

1.3 Metodologias

O desenvolvimento deste projeto possui características específicas que sugerem que o seu próprio paradigma de investigação passa por um método mais próximo com o público-alvo com testes num ambiente iterativo de desenvolvimento da solução.

Uma categorização proposta num workshop em Dagstuhl [THP93], agrupa os métodos de investigação em quatro categorias gerais:

[5] http://www.dge.mec.pt/curso-de-ciencias-e-tecnologias-0

1. Método Científico: "Cientistas desenvolvem uma teoria para explicar o fenómeno; Estes propõem uma hipótese e testam depois alternativas variadas da hipótese. Enquanto o fazem, recolhem dados para verificar ou refutar as afirmações da hipótese."
2. Método de Engenharia: "Os engenheiros desenvolvem e testam a solução a uma hipótese. Com base nos resultados do teste, estes melhoram a solução até não serem necessários mais melhoramentos."
3. Método Empírico: "Um método estatístico é proposto como meio de validar uma hipótese dada. Ao contrario do método cientifico, poderá não existir um modelo ou teoria formal para descrever a hipótese. Dados são recolhidos para verificar a hipótese."
4. Método Analítico: "Uma teoria formal é desenvolvida e os resultados que derivam dessa teoria podem ser comparados com observações empíricas."

Estas categorias aplicam-se para a ciência em geral, nesta dissertação optar-se-á por métodos de engenharia.

1.3.1 Modelo de Investigação e Fases

A estratégia para este projeto prende-se com o objeto de investigação, os resultados a produzir (uma hipótese,um modelo, um processo ou um produto), o propósito (caracterização, avaliação, predição, controlo, melhoramento), o foco (o aspeto de interesse ou objeto de estudo) e os recursos disponíveis (tempo, projetos, hardware).

De modo a uma aprendizagem com sucesso de uma linguagem de programação Du Bouley et al [DBOM99] estabeleceram as seguintes condições:

1. Uma máquina de notação concetual simples
2. Alguns destes processos desta máquina devem ser visíveis para o utilizador
3. O sistema deverá ser interativo
4. Os materiais de ensino usados deverão ser harmoniosos com a implementação particular da linguagem
5. Os comentários deverão ser de um nível apropriado de detalhe para a tarefa atribuída e para o sua perceção concetual.

Tendo em consideração o enquadramento e objetivos desta dissertação, foi seguido um método dentro da categoria dos métodos de engenharia, onde são inicialmente selecionadas as várias ferramentas e estratégias para a solução geral, estas são depois integradas numa solução particular de acordo com o público-alvo em questão, sendo posteriormente testadas em atividades de campo periódicas. De seguida é recolhido o *feedback* dos participantes em cada atividade de campo, ajudando assim no desenvolvimento das várias iterações do projeto, até chegar a uma solução que preencha as lacunas que se propõe suprimir.

Usaram-se também métodos empíricos com alguns dados recolhidos durante as experiências, com recurso a alguns inquéritos dirigidos aos alunos sobre fatores específicos da ação assim como sessões informais de partilha de experiências com o público-alvo durante as atividades de campo. Estes dois métodos foram usados em conjunto com a validação das várias instâncias de todo o conteúdo desenvolvido, com recurso a experiências de campo organizadas ao longo da dissertação.

1.4 Contribuições

Neste contexto, esta dissertação tem como objetivo fundamental promover e facilitar o ensino da programação a jovens, envolvendo respetivos professores e encarregados de educação. Em concreto, pretende-se contribuir para o estado da arte do ensino da programação em Portugal ao:

- Criar e divulgar conteúdos e atividades de programação para jovens em duas faixas etárias distintas, dos 12 aos 14 e dos 15 aos 18 anos;
- Definir estratégias e selecionar ferramentas motivadoras para o ensino de programação para jovens, com base em boas práticas de desenvolvimento de software.

Estas contribuições estão prontas para ser postas em prática em várias atividades piloto no Coding for Social Impact Labs[6] da FEUP, onde desempenho a função de *Tech Leader*. Note-se, no entanto, que uma das atividades[7], foi já apresentada em Dezembro de 2016, no 1º Movimento Código Portugal, uma iniciativa do Ministério da Educação de Portugal que ocorreu no Pavilhão do Conhecimento, em Lisboa.

1.5 Estrutura da Dissertação

Este documento é composto por cinco capítulos. No capítulo 2 são explicados os principais conceitos de tecnologia educativa (*Edutech*) e feito o levantamento do estado da arte tecnologias educativas desenvolvidas para o ensino da programação. O capítulo 3 descreve a solução proposta para dinamizar ensino da programação para jovens em Portugal, sendo que as atividades desenvolvidas para o efeito são detalhadas no capítulo 4. O capítulo 5 apresenta as principais conclusões do trabalho desenvolvido, referindo o trabalho futuro a desenvolver nesta área de investigação.

[6] http://codingforsocialimpact.org
[7] https://www.codemove.pt/uploads/programadia11.pdf

Capítulo 2

Tecnologia Educativa para o Ensino da Programação

A tecnologia educativa (*Edutech*) é um campo bastante vasto, o que por consequência gera várias definições que podem ser conflituosas entre si [1]. No entanto, vários autores concordam que:

1. O uso da tecnologia é um princípio da prática da Edutech: a tecnologia educativa é baseada em conhecimento teórico de outras áreas (comunicação, educação, ciência, etc.).

2. Edutech tem como objetivo melhorar a educação: a tecnologia deve facilitar o processo de aprendizagem e melhorar a performance de sistemas educativos, no que diz respeito à eficácia e/ou eficiência.

2.1 Edutech

A investigação da Edutech sempre teve uma agenda ambígua: por vezes concentra-se apenas em aumentar a eficiência ou eficácia de práticas correntes, mas tem também frequentemente como foco mudanças pedagógicas. Apesar de ser considerada como uma ciência de design, também se foca nos problemas de aprendizagem, ensino e organização social, pelo que faz uso de todo o espectro da ciência social moderna e metodologias de ciências da vida.

A tecnologia é então tanto uma ferramenta, como um catalizador, podendo tornar-se num meio em que podem ocorrer mudanças.

Existem várias perspetivas sobre a Edutech:

1. Perspetiva de Design Instrucional: além de um campo de investigação, Edutech é sinónimo de pedagogia e aprendizagem com tecnologia e, como tal, uma disciplina de engenharia, uma ciência de design ou uma arte.

[1] $http://edutechwiki.unige.ch/en/Mainpage$

2. Perspetiva Orientada à Investigação de Projetos: onde se adota uma abordagem interdisciplinar que levará ultimamente a melhores projetos pedagógicos numa área específica.

3. Perspetiva de Investigação Fundamental: muitos investigadores adotam uma perspetiva mais focada em assuntos particulares (por exemplo: em que condições podem as animações multimédia ser eficazes).

4. Perspetiva Institucional: uma área é implicitamente definida por jornais, conferências e programas de estudo. O Journal of Interactive Learning Research [Ass97] (Jornal de Investigação de Aprendizagem Interativa) publicado pela associação de Advancement of Computing in Education (Avanços da Computação na Educação) listou as várias áreas técnicas da Edutech: Sistemas de Escrita de Conteúdos, Ferramentas Cognitivas para Aprendizagem de Linguagens Assistidas por Computador, Sistemas de Avaliação Baseados em Computadores, Treino Baseado em Computador, Comunicações Mediadas por Computador, Aprendizagem Colaborativa Suportadas por Computador, Ambientes de Aprendizagem Distribuída, Sistemas de Suporte para Performance Eletrónica, Ambientes de Aprendizagem interativa, Sistemas Multimédia Interativos, Simulações Interativas e Jogos, Agentes inteligentes na Internet, Sistemas de Ensino Inteligentes, Micromundos e Sistemas Baseados em Realidade Virtual.

5. Perspetiva Tecnológica: pode-se afirmar que a investigação e prática da Edutech está intimamente relacionada com o desenvolvimento tecnológico [Cha95].

6. Perspetiva "Onde é usada"onde podemos identificar: Educação à Distância, B-Learning, Salas de Aula Melhoradas por Tecnologia e Ensino Informal.

Pode-se então afirmar que, como podemos visualizar na Figura 2.1, a Edutech é um todo estruturado. Uma das áreas sobre a qual a Edutech se foca é no ensino da programação, havendo já um grande leque de ferramentas disponíveis para este fim.

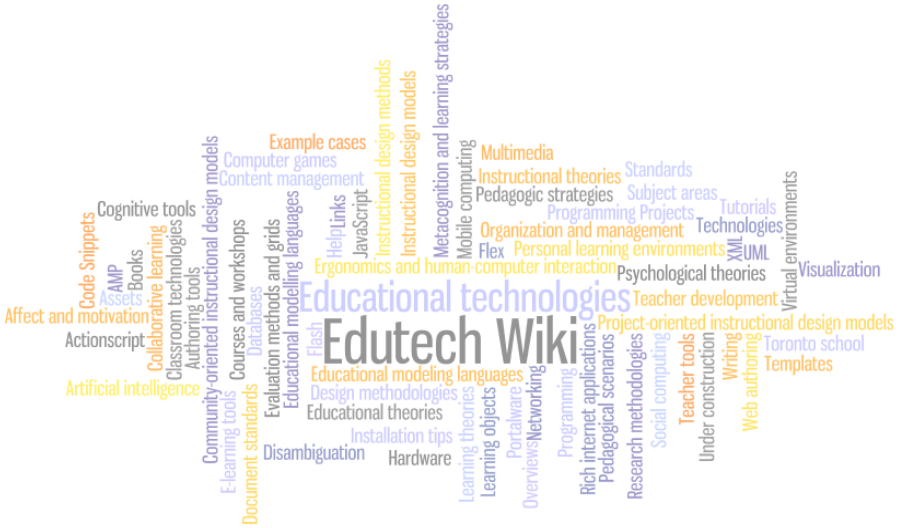


Figura 2.1: EduTech Wiki: Um todo estruturado

2.2 Ensino da Programação

Nesta secção apresentam-se algumas das tecnologias mais utilizadas no ensino da programação a jovens em todo o mundo.

2.2.1 Alice

Alice é um ambiente de programação 3D que facilita a criação de animações para contar uma história, jogar um um jogo interativo, ou um vídeo para partilhar na *web*. Esta ferramenta de ensino está disponível de modo gratuito e foi concebida para ser a primeira exposição de programação orientada a objetos para os estudantes [PEH07]. Permite a aprendizagem de conceitos fundamentais de programação no contexto de criação de filmes de animação e jogos de computador simples. Nesta ferramenta, objetos 3D (pessoas, animais, veículos, etc.) povoam um mundo virtual e os estudantes criam um programa para animar os objetos. Nesta ferramenta, os estudantes arrastam e largam blocos gráficos para criar um programa, onde as instruções correspondem a declarações *standard* numa linguagem de programação *production oriented*, como Java, C++ e C#. *Alice* permite que os estudantes vejam de imediato como é que irão correr as suas animações, permitindo-lhes perceber facilmente a relação entre a programação e o comportamento de objetos nas suas animações. Ao manipular os objetos nos seus mundos virtuais, os estudantes ganham experiência com todas as construções de programação tipicamente ensinadas num curso introdutório de programação.

Figura 2.2: Logótipo Alice

2.2.2 App Inventor

O *App Inventor* é uma aplicação *web* que permite desenhar aplicações Android e corrê-las posteriormente em dispositivos Android. A aplicação fornece um emulador de Android de modo a que os participantes não necessitem de um dispositivo físico para correr as suas aplicações. Normalmente apresentam-se os seguintes conceitos nas atividades relacionadas com esta ferramenta: programação *event-driven*, criação de variáveis, declarações de seleção e repetição, geração de aleatórios, trabalhar com ficheiros de imagem e de som e funções.

Figura 2.3: Logótipo App Inventor

2.2.3 Scratch

O *Scratch* [RSK+09] é um ambiente de programação visual com bastante popularidade desenhado para ensinar conceitos de programação a estudantes, fazendo uso de uma metáfora de blocos onde o "código"é criado ligando blocos entre si. Com esta ferramenta possibilita-se aos estudantes que criem as suas próprias histórias e que as partilhem com colegas na comunidade online do *Scratch*. Trata-se de um projeto do *Lifelong Kindergarten Group* do *MIT Media Lab* e

é uma ferramenta completamente gratuita. O *Scratch* foi concebido para jovens dos oito anos de idade aos dezasseis, apesar de não excluir pessoas com outras idades.

Normalmente apresentam-se os seguintes conceitos nas atividades relacionadas com esta ferramenta: tomada de decisões com blocos *if then else*, operadores Booleanos, coordenadas X e Y, variáveis, temporizadores, transmissão e receção de sinais, mudanças de vestimenta de personagens, importação de sons e *sprites*, esperas e mostrar/esconder elementos.

Figura 2.4: Logótipo Scratch

2.2.4 Snap!

Snap! [2] é uma variante baseada no código fonte do *Scratch*, permite que se criem os seus próprios blocos de código para serem usados. Devido à sua implementação em *Javascript*, esta ferramenta corre num navegador de internet. Possui todas as funcionalidade do *Scratch*, adicionando-se *first class lists* (podemos armazenar listas em variáveis, passar listas para funções e devolver listas em funções), *lambdas* (uma expressão cujo valor é uma função) e predicados customizados. Normalmente apresentam-se os seguintes conceitos nas atividades relacionadas com esta ferramenta: repetição, estruturas, operadores de *strings*, *scripts* em vez de variáveis globais, blocos customizados e operações em listas.

Figura 2.5: Logótipo Snap!

[2] http://snap.berkeley.edu/

2.2.5 Python

Python é uma linguagem de muito alto nível, fácil de aprender que possui características bastante interessantes para quem se está a iniciar na programação. O nome desta linguagem de programação vem do programa de televisão *Monty Python's Flying Circus*.

Concentra experiência de vários anos de trabalho em ciência de computação e incorpora ideias que se encontram em paradigmas de programação imperativos, orientados a objetos e funcionais. Inclui exceções, módulos e classes.

O código é de fácil leitura, quando comparado com outras linguagens de programação (C, C++, Java...), e possui uma *shell* interativa que permite testar código diretamente e visualizar as aplicações a correr.

Em *Python* o modelo de objeto é de fácil utilização, é interpretado, e de modo a maximizar a velocidade de desenvolvimento, sacrifica alguma performance ao ser simplesmente interpretado. Pode chamar outros programas e pode ser embebido noutros programas. *Python*, ao contrário de outras linguagens de aprendizagem de programação, não é considerada uma linguagem "brinquedo".

Figura 2.6: Logótipo Python

2.2.6 Cursos Abertos Massivos Online (*MOOCs*)

Muitos professores de educação de tecnologia nos K12 (do ensino primário ao ensino secundário, inclusive) pretendem integrar o ensino da programação nos planos das suas aulas, mas não têm bem a noção por onde começar. [Kol13] Várias universidades nos Estados Unidos, em conjunto com parceiros da indústria, já oferecem *workshops* presenciais ou disponibilizam conteúdos *online* para ajudar e treinar a comunidade de K12 a integrar o ensino da programação nos seus planos de aulas. Alguns educadores apontam os *MOOCs* como uma ferramenta alternativa para as formações necessárias à comunidade K12.

Um MOOC consiste num curso *online* em larga escala oferecido a audiências abertas. Tem como objetivo oferecer educação de alta qualidade a plateias de grandes dimensões, fazendo uso de recursos educacionais de acesso livre e a baixo custo.

Muitas universidade como Harvard, Stanford e o MIT oferecem MOOCs em plataformas como o Coursera e o edX. Estes cursos já tiveram entre os dez mil e os cento e cinquenta mil, ou mais, participantes registados, fazendo uso frequentemente de vídeos, tutoriais e exercícios numa experiência construtiva.

Cursos mais recentes lecionados em MOOCs têm-se focado em grupos colaborativos para engenharia de *software*.

Um estudo de 2013 em *Chronicle of Higher Education* revelou que 79% dos cento e três professores que se encontravam a lecionar MOOCs, gastavam em média cem horas para preparar o seu MOOC. Da média de 33000 estudantes que participavam nos seus MOOCs, cerca de 2600 estudantes (7.8%) completaram o curso com uma nota positiva [Kol13].

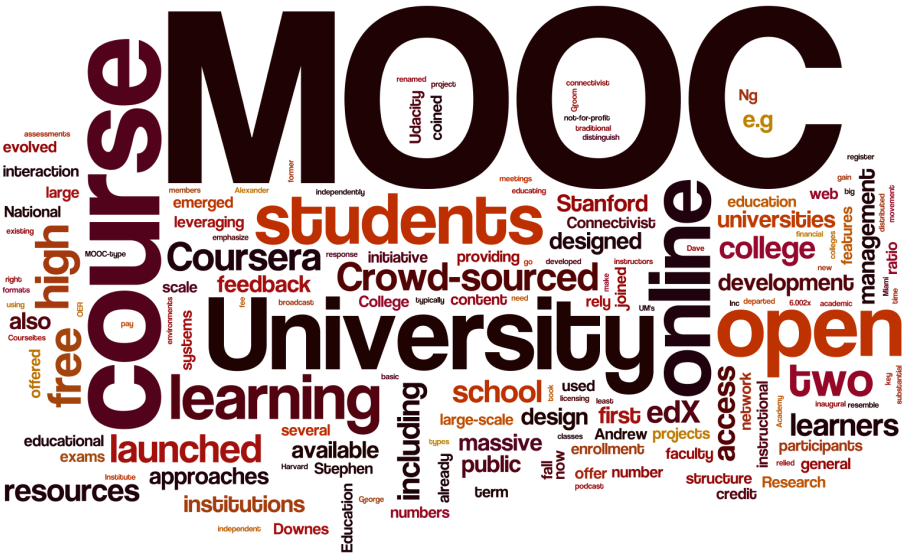


Figura 2.7: Word Bubble de MOOCs

2.2.7 Open edX

O *edX* é uma iniciativa online sem fins lucrativos criada por parceiros de Harvard e MIT. É composto por dezenas de instituições líder globais que fazem parte do *xConsortium*. O *edX* oferece interatividade em cursos online e MOOCs das melhores universidades e instituições do mundo.

O Open edX é a plataforma *open-source* que está na base dos cursos *edX*. Os programadores podem contribuir livremente para esta plataforma desenvolvendo novas funcionalidades e cursos.

Figura 2.8: Logótipo OpenEDX

2.2.8 GitHub educação

O GitHub é um sistema Git (um sistema de controlo de versões para ficheiros digitais) baseado em web com serviço de hospedagem na Internet. Oferece o controlo de versões distribuído e gestão de código-fonte, bem como adiciona ainda os seus próprios recursos, como por exemplo o controlo de acessos e vários recursos de colaboração, como rastreamento de *bugs*, solicitações de recursos, gestão de tarefas e wikis para cada projeto. Esta plataforma pode ser livremente integrada noutras soluções de modo a permitir o desenvolvimento de projetos em grupo, de modo completamente colaborativo, isto com recurso a todos os métodos comuns a um projeto partilhado via Git.

O GitHub educação [Git17] é um programa lançado pelo GitHub, dedicado a estudantes e educadores, que permite que alunos de todo o mundo usem repositórios de código Git gratuitamente. Evita-se deste modo que os alunos enviem código para os seus colegas via e-mail, o que aliado às funcionalidades para mostrar as alterações e para juntar código, os alunos dificilmente terão que se preocupar em apagar por engano o código dos seus colegas.

Figura 2.9: Logótipo GitHub

2.2.9 Ace

Este editor web *open-source* para código permite, de raiz, a integração com outras soluções. Com uma enorme facilidade possibilita também que se desenvolvam novas funcionalidades por intermédio de *plugins*, para novas linguagens, automatizações, correções, etc.

Figura 2.10: Logótipo Ace

2.2.10 Raspberry Pi

O *Raspberry Pi* [3] é um computador de bolso, de baixo custo, com o tamanho de um cartão de crédito, que pode ser ligado a um monitor ou a uma televisão e usa um teclado e rato convencional. Trata-se de um dispositivo que possibilita a pessoas de todas as faixas etárias aprenderem programação recorrendo a linguagens de programação como *Scratch* ou *Python*. É capaz de executar praticamente todas as tarefas que se espera que um computador execute, desde navegar na Internet ou reproduzir filmes em alta definição, criar folhas de cálculo, documentos e jogar.

O RPi continua a ser usado por entusiastas de todo o mundo para os mais variados projetos de eletrónica e programação e a *Raspberry pi Foundation*, tem como objetivo que o RPi seja usado por crianças de todo o mundo para aprenderem programação e perceberem como funcionam os computadores.

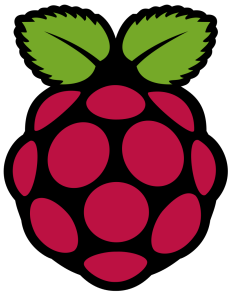


Figura 2.11: Logótipo Raspberry Pi

2.3 Iniciativas

Neste momento estão a decorrer várias iniciativas de vários parceiros em todo o globo de promoção da programação para jovens.

2.3.1 Khan Academy

Khan Academy [4] disponibiliza vários recursos para todas as idades, onde são oferecidos exercícios práticos e vídeos instrucionais aliados a uma aprendizagem que permite vários ritmos dentro e fora da sala de aula. São disponibilizadas, entre outras, várias áreas como: matemática, ciência, programação, história e economia.

Esta iniciativa possui parcerias com a *NASA*, *The Museum of Modern Art* dos EUA, *California Academy of Sciences* e o *MIT*, com oferta de conteúdo especializado. Todo o conteúdo disponibilizado de um modo gratuito, tanto para pais como para professores. Os recursos estão atualmente a ser traduzidas para mais de trinta e seis línguas.

[3] http://raspberrypi.org
[4] http://www.khanacademy.org/

Figura 2.12: Logótipo Khan Academy

2.3.2 Code.org

A iniciativa sem fins lucrativos *Code.org* [5] foi lançada em 2013 e dedica-se à expansão na participação em ciência de computação ao torná-la disponível em mais escolas e aumentando a participação de mulheres e estudantes de minorias, com a visão de que cada estudante em cada escola deverá ter a oportunidade de aprender ciência de computação. Acreditam que o ensino de programação deverá ser parte do núcleo do programa escolar, ao lado doutras ciências, tecnologias, engenharias, biologia, física, química e álgebra.

No cursos disponibilizados, 43% dos estudantes são raparigas e 37% são de minorias, ao passo que nas salas de aula dos Estados Unidos da América, 34% são raparigas e 60% são minorias.

Dezenas de milhões de alunos já experimentaram o *Hour of Code* (123.205.924 estudantes),48% são mulheres. 99% dos professores questionados recomendam o *Code.org* para introdução à ciência de computação. 144.241 professores inscreveram-se para ensinar os cursos de introdução no *Code Studio* e 5.972.864 estudantes estão inscritos. Prepararam mais de 9000 professores para ensinar ciência de computação em vários anos K12. Possuem parceiras com mais de 70 dos maiores distritos escolares para adicionarem ciência de computação ao seu programa. Os seus cursos estão disponíveis em mais de 30 línguas e é usado em mais de 180 países.

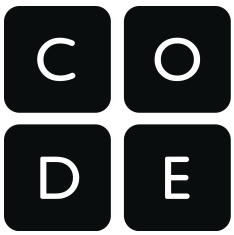


Figura 2.13: Logótipo Code.org

2.3.3 Google Computer Science First

O Google CS First [6] possibilita a todos os estudantes a criação de tecnologia a partir de grupos de ciência de computação gratuitos, com uma grande variedade de conteúdos disponibilizados de modo a motivar uma população estudante bastante diversa.

[5] https://code.org/about
[6] https://www.cs-first.com

Estão disponíveis grupos de computação temáticos baseados em temas do mundo real com ofertas de 10 horas de lições e atividades. Estes grupos com temas diversos têm como objetivo atrair e motivar estudantes com e experiências diferentes.

Todos os materiais são destinados a estudantes do quarto ao oitavo ano (ou entre as idades de nove e catorze anos), são gratuitos e de fácil utilização.

Não é necessária experiência de ciência de computação para ser anfitrião de um grupo *CS First*. Apenas é necessário o acesso a uma localização, um computador por participante e um grupo de estudantes.

Figura 2.14: Logótipo Google CS First

2.3.4 O caso de Portugal

Em Portugal, o Ministério da Educação e Ciência anunciou recentemente a integração do ensino da programação no ensino básico [7], onde estão inscritos 240 agrupamentos de escolas, 1805 turmas, contemplando a formação de 720 professores [8]. Esta iniciativa iniciação à programação iniciou-se no ano letivo 2015/2016 [9].

Foi ainda aprovada a criação do Conselho para as Tecnologias de Informação e Comunicação que irá "assegurar o desenvolvimento de uma estratégia global de planeamento e otimização das TIC na Administração Pública".

Têm, ainda, decorrido algumas iniciativas individuais [10], em Lisboa com o apoio financeiro de 120 mil euros [11] [12], alguns programas para programação nas férias [13], Scratch Day [14] ou contribuições para aulas com suporte em B-learning [15] [16].

Movimento Código Portugal[17] é uma campanha nacional de consciencialização com foco na literacia digital e computacional.

[7] http://wintech.pt/44-noticias/noticias-gerais/18932-ministerio-confirma-integracao-de-aulas-de-programacao-no-ensino-basico

[8] http://pplware.sapo.pt/informacao/aulas-de-programacao-vao-chegar-ao-1o-ciclo-do-ensino-basico/

[9] http://www.erte.dge.mec.pt/iniciacao-programacao-no-1o-ciclo-do-ensino-basico

[10] http://wintech.pt/120-noticias/empreendedorismo/20948-investigadores-portugueses-testam-projeto-que-permite-educar-alunas-atraves-de-jogos

[11] http://www.publico.pt/local/noticia/camara-de-lisboa-e-gulbenkian-levam-programacao-informatica-as-escolas-1685304

[12] http://www.academiadecodigo.org/

[13] http://wintech.pt/120-noticias/empreendedorismo/18689-rumos-anuncia-informatica-jovem-para-ocupacao-de-ferias-da-pascoa

[14] http://wintech.pt/44-noticias/noticias-gerais/18937-instituto-politecnico-de-setubal-recebe-scratch-day-2015

[15] http://wintech.pt/49-noticias/isp-s/18873-fundacao-pt-ultrapassa-1000-videos-na-khan-academy

[16] http://wintech.pt/120-noticias/empreendedorismo/18591-docentes-do-politecnico-de-setubal-distinguidos-pela-fundacao-para-a-ciencia-e-a-tecnologia

[17] https://www.codemove.pt/

O Movimento Código Portugal é uma iniciativa do Governo Português, através de um conglomerado das áreas da Ciência, Tecnologia e Ensino Superior, Educação, Economia, Trabalho, Solidariedade e Segurança, a Ciência Viva, Agência Nacional para a Cultura Científica e Tecnológica, e a Faculdade de Ciências e Tecnologia da Universidade NOVA de Lisboa.

Este Movimento pretende estimular o desenvolvimento das competências associadas ao pensamento computacional, com a justificação de que se trata da nova forma de literacia deste século.

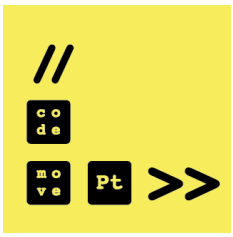


Figura 2.15: Logótipo Movimento Codigo Portugal

2.3.5 Investimento

Pode-se ainda aferir parte do sucesso das iniciativas relacionadas com Edutech se analisarmos as sucessivas entradas de capital nas suas empresas nos últimos anos, de onde se destacam mais recentemente as seguintes, por ordem de grandeza decrescente em milhões (M) de euros:

Age of Learning (141M€[18]), BYJU (70.5M€[19]), Udemy (60M€[20]), Coursera (47M€[21]), Brainly (14M€[22]), FreshGrade (10.9M€[23]), Tynker (6.7M€[24]), Nasscom Foundation (4.5M€[25]), pi-top (4M€[26]), Code Spark (3.86M€[27]), Teachable (3.74M€[28], LocoMotive Labs (3.7M€[29]), Duck Duck Moose (2.8M€[30]), Piper (1.97M€[31]) e CodeCombat (1.9M€[32]).

[18] https://techcrunch.com/2016/05/03/age-of-learning-a-quiet-giant-in-education-apps-raised-150m-at-a-1b-valuation-from-iconiq/
[19] https://techcrunch.com/2016/03/21/ed-tech-startup-byju-raises-75m-from-sequoia-india-and-sofina/
[20] https://techcrunch.com/2016/06/02/edtech-startup-udemy-raises-60-million-from-naspers-for-international-expansion/
[21] https://techcrunch.com/2016/08/31/coursera-expands-into-corporate-learning-and-development
[22] https://www.edsurge.com/news/2016-05-13-brainly-raises-15-million-series-b-to-expand-q-a-platform-beyond-homework-help
[23] https://www.edsurge.com/news/2016-05-10-student-portfolio-startup-freshgrade-gets-an-a-along-with-11-6-million-in-funding
[24] https://techcrunch.com/2016/05/31/tynker-raises-7-1m-to-expand-its-code-teaching-programs-to-new-schools-and-regions/
[25] https://techcrunch.com/2016/03/09/nasscom-foundation-gets-4-78m-from-the-gates-foundation-to-support-tech-programs-in-indian-public-libraries
[26] https://techcrunch.com/2016/11/03/learn-to-code-startup-pi-top-pulls-in-4-3m-to-fund-a-global-edtech-push/
[27] https://techcrunch.com/2016/09/29/codespark-raises-4-1-million-for-games-that-teach-kids-how-to-code/
[28] https://techcrunch.com/2017/01/25/teachable-books-4-million-to-turn-everybody-into-educators-online/
[29] https://techcrunch.com/2015/02/26/locomotive-labs-series-a/
[30] https://techcrunch.com/2016/08/26/kids-app-maker-duck-duck-moose-joins-khan-academy/
[31] https://techcrunch.com/2016/06/21/piper-seed-round/
[32] https://www.edsurge.com/news/2016-05-02-codecombat-scores-2-million-seed-round-to-bring-programming-adventure-to-schools

Estes exemplos demonstram a tração que as empresas de Edutech têm ganho no mercado sendo um assunto bastante apetecível para o desenvolvimento dos "programadores"do futuro, de modo a sucumbir as necessidades prementes em especialistas tecnológicos na indústria. Nos Estados Unidos da América Tim Cook (CEO Apple), Mark Zuckerberg e Sheryl Sandberg (CEO Facebook e COO Facebook respetivamente) e Doug McMillon (CEO Walmart) pediram ao Congresso Americano que financiasse com 250 milhões de dólares os distritos escolares de modo a que todas as crianças no país tivessem a oportunidade de aprender a programar [33]. Entretanto Microsoft, Google, Zuckerberg, Bezos e outros comprometeram-se coletivamente com 48 milhões de dólares para a causa, de onde 23 milhões foram destinados para o *Code.org*.

2.4 Resumo

Pode-se concluir que a Edutech é um tema de crescente importância e que tem vindo a aumentar o seu relevo e projeção na comunicação social, nomeadamente em Portugal, quer por parcerias governamentais, que deram origem à sua implementação no ensino básico, quer pelo financiamento de vários projetos de incentivo de ensino à programação. Não obstante, é expectável que hajam alguns reajustamentos ao plano do governo português. Foram ainda apresentadas várias ferramentas atualmente usadas no ensino de programação em K12, assim como as principais iniciativas a decorrer em todo o globo de promoção da programação para K12.

[33] https://techcrunch.com/2016/04/26/tim-cook-mark-zuckerberg-and-others-urge-congress-to-fund-k-12-computer-science-education/

Capítulo 3

Ensino de programação a jovens

O ensino de programação a jovens tem-se tornado num dos assuntos mais debatidos em revisões curriculares do sistema de ensino de alguns países. A literatura sobre este assunto é vasto estendendo-se às últimas 3 décadas, incluindo linguagens como o LOGO ou o BASIC nos anos 80, que endereçam o problema de ensino e aprendizagem de programação e Ciência da Computação [GP13].

3.1 Problema

O problema identificado passa pela necessidade de transmissão de conhecimentos de programação a jovens portugueses com conteúdos e atividades relevantes, que mais do que ensinarem a programar, ensinem a pensar de um modo estruturado e colaborativo, como é necessário em ciência de computação [JSS04]. Surge também a necessidade de formar os educadores para o correto ensino da problemática [EGM14], assim como a transmissão de conhecimentos de um modo divertido, e estimulante, para que o ensino de programação não seja encarado pelos alunos como mais uma disciplina.

3.2 Oferta educativa

A oferta curricular de programação, ou mais genericamente referida como pensamento computacional (quando nos referimos ao ensino obrigatório), e de acordo com um estudo da Comissão Europeia de 2016 [BCD+16], é referida inúmeras vezes como a aptidão fulcral do século 21. Esta aptidão deverá capacitar os alunos para serem tecnologicamente literatos, mas também para serem capazes de criar artefactos computacionais.

3.2.1 Unidades didáticas

Neste contexto, e face à escassez de conteúdos em português indicados para o ensino de programação a jovens, desenharam-se e conceberam-se várias unidades didáticas que utilizam software, hardware e estratégias pedagógicas adequadas para o ensino da programação. Estas unidades didáticas figuram como exemplos de uma oferta educativa mais abrangente, sendo que a natureza e tipo do software, hardware, e estratégia de ensino a seguir, são o principal foco do trabalho desenvolvido nesta dissertação.

As várias unidades didáticas foram idealizadas de modo a explorar os vários projetos e conceitos de programação desenvolvidos. Cada unidade didática subdivide-se em:

1. Título da atividade;
2. Subtítulo onde é descrita a atividade numa frase;
3. Objetivos pedagógicos que se pretendem alcançar no final da unidade didática;
4. Duração da atividade em minutos;
5. Material necessário para percorrer a atividade e o seu preço unitário;
6. Instruções passo a passo;
7. Próximas atividades onde se sugerem alternativas de percurso;
8. Referências para a unidade didática em questão.

Estas unidades didáticas são ainda acompanhadas dum glossário de apoio, por existirem muitos conceitos transversais, referenciados múltiplas vezes entre unidades.

3.2.2 Principais conceitos de programação

> "O pensamento computacional é uma metodologia de resolução de problemas que expande o domínio da informática em todas as disciplinas, fornecendo um meio distinto de analisar e desenvolver soluções para problemas que podem ser resolvidos computacionalmente. Com foco na abstração, automação e análise, o pensamento computacional é um elemento central da disciplina mais ampla da ciência da computação."
>
> — Computer Science Teachers Association, CSTA K–12 Computer Science Standards.[CST12]

Se partirmos para os principais conceitos e definições do pensamento computacional encontramos:

- Abstração é o processo de tornar um artefacto mais compreensível através da redução dos detalhes desnecessários. A habilidade em abstração está em escolher o detalhe certo

para esconder para que o problema se torne mais fácil, sem perder nada que seja importante [CCD+15].

- Decomposição é um modo de pensar sobre artefactos a nível dos seus componentes. As partes podem então ser compreendidas, resolvidas, desenvolvidas e avaliadas separadamente. O que torna os problemas complexos mais fáceis de serem resolvidos [CCD+15].

- Automação é um processo de poupança de trabalho onde um computador é instruído de executar um conjunto de tarefas repetitivas de forma rápida e eficiente em comparação com o poder de processamento de um ser humano. Nesta perspetiva, os programas são automações de abstrações [LMD+11].

- O pensamento algorítmico é uma maneira de chegar a uma solução através de uma definição de pensamento clara dos passos necessários para resolver determinado problema [CCD+15].

- A generalização está associada à identificação de padrões, semelhanças e conexões, e à exploração desses recursos. Fazem-se perguntas como "Isto é semelhante a um problema que eu já resolvi?"e "Em que difere?" [CCD+15].

- A depuração é a aplicação sistemática de análise e avaliação usando habilidades como testes, rastreamento e pensamento lógico para prever e verificar resultados [CCD+15].

A nível de conceitos transversais de linguagens de programação optaram-se pelos seguintes, por serem comuns a grande parte das linguagens: instruções, funções, variáveis, condicionais, ciclos e bibliotecas.

3.2.3 Para além do código

Mais do que criar outra ferramenta de ensino da programação, o foco desta dissertação é criar uma solução integradora que transmite metodologias de desenvolvimento de software utilizadas por profissionais da área, aproximando os alunos da realidade e evidenciando o papel da programação como uma ferramenta para o desenvolvimento de software. Assim, pretendeu-se selecionar ainda algumas ferramentas de desenvolvimento ágil de software, nomeadamente o *Scrum*, e simplificar alguns conceitos de modo a serem aplicáveis ao público-alvo.

De acordo com a Scrum Alliance [Scr11] o *Scrum* pode ser descrito em 30 segundos do seguinte modo: "Um proprietário do produto cria uma lista de desejos priorizada chamada de *backlog* de produto. Durante o planeamento do *sprint*, a equipa seleciona um pequeno pedaço do topo da lista de desejos, um *backlog* de *sprint*, e decide como implementar essas peças. A equipa tem uma certa quantidade de tempo - um *sprint* (geralmente uma a quatro semanas) - para completar o seu trabalho, mas que se reúne a cada dia para avaliar o seu progresso (*Scrum* diário). Ao longo do caminho, o ScrumMaster mantém a equipa focada no seu objetivo. No final do *sprint*, o trabalho deve ser potencialmente enviado: pronto para entregar a um cliente, colocar numa prateleira de uma loja, ou mostrar a um *stakeholder*. O *sprint* termina com uma revisão de *sprint* e retrospetiva.

À medida que o próximo *sprint* começa, a equipa escolhe outro pedaço da lista de tarefas e começa a trabalhar novamente."

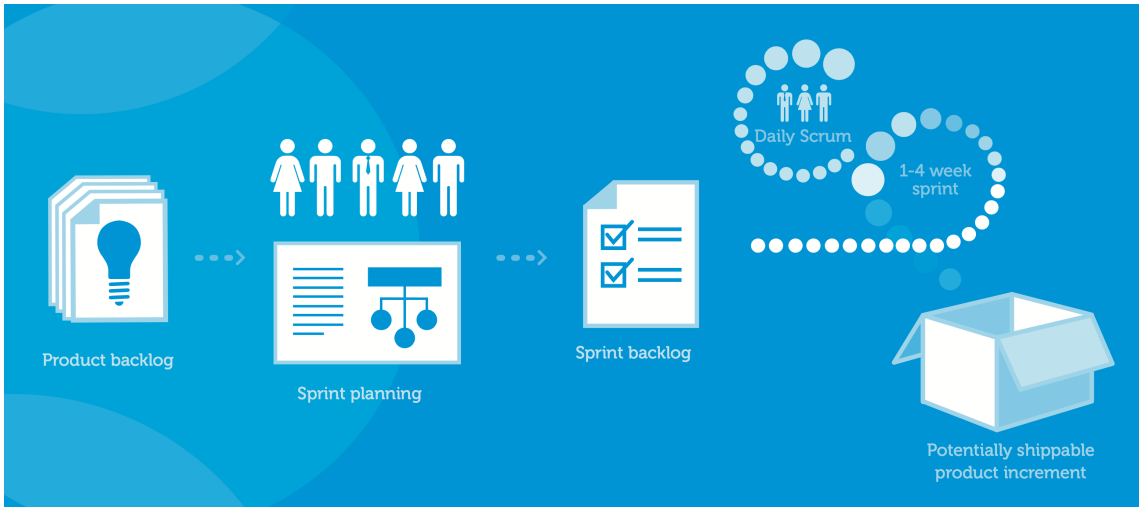


Figura 3.1: Resumo da *framework* do Scrum [All]

O uso de Scrum na sala de aula foi já idealizado [RN16] o que, aliado ao seu uso generalizado na indústria do desenvolvimento e software, motivou a sua integração na solução a desenvolver.

3.3 Estratégias Pedagógicas

A estratégia pedagógica adotada numa oferta educativa pode condicionar o sucesso da mesma, devendo ser escolhida tendo em consideração o público-alvo e o tipo de conhecimentos que se pretendem transmitir. As próximas secções descrevem os tipos de estratégias pedagógicas seguidas, ressalvando as caraterísticas mais distintivas que as tornam adequadas para o ensino de programação a jovens.

3.3.1 Aprendizagem baseada em projetos

O PBL - *Project Based Learning* é definida por Thomas [Tho00] como "a aprendizagem que é baseada em projetos"tem o poder para motivar e entusiasmar, mas também tem suas desvantagens na escolaridade formal.

Thomas lista os atributos de PBL como:

- tarefas complexas, baseadas em questões ou problemas desafiadores, que envolvem alunos em projetos, resolução de problemas, tomada de decisões ou atividades de investigação;
- dar aos alunos a oportunidade de trabalhar de forma relativamente autónoma durante longos períodos de tempo;
- e culminando em produtos ou apresentações realistas.

Outras características encontradas no mesmo artigo e de igual importância incluem: facilitação do professor (mas não orientação), objetivos educacionais explícitos, aprendizagem cooperativa, reflexão e incorporação de habilidades de adultos.

Todos estes argumentos levam a que PBL seja um candidato ideal a estratégia principal de ensino.

3.3.2 Programação em pares

A programação em pares consiste em dois programadores a partilhar um único computador entre eles (um ecrã, um teclado e um rato). O programador no teclado é geralmente chamado de condutor (*driver*), o outro também envolvido ativamente na tarefa de programação, mas focando mais na direção geral é o navegador (*navigator*). Espera-se que os programadores invertam os papéis a cada poucos minutos.

Algumas das principais vantagens desta prática são:

- Resiliência a interrupções melhorada, em comparação com um programador individual: quando um membro do par tem uma solicitação externa, o outro pode permanece focado na tarefa e pode ajudar a recuperar o foco depois;
- Melhor difusão do conhecimento entre o grupo, em particular quando um programador não familiarizado com um componente está emparelhado com alguém que sabe muito mais;
- Maior qualidade de código: "programar em voz alta"leva a uma articulação mais clara das complexidades e detalhes nas tarefas de programação, reduzindo o risco de erros.

Quando aplicados a alunos estudos demonstram que a programação em pares contribuí para a redução de frustração e os ajuda a progredir com alguma distinção a nível de notas [WWY02].

3.4 Software

Relativamente ao software a integrar na solução a desenvolver, vários candidatos foram considerados, tendo a escolha recaído, sempre, por opções *Open-Source* [1], gratuitas e com uma comunidade estabelecida e em expansão. Como tal, idealiza-se uma solução integradora com as ferramentas em baixo descritas.

3.4.1 Scratch

O Scratch foi escolhido para ser parte integrante das nesta dissertação, por ser possível estabelecer desafios que os jovens consideram interessantes, sem recuso à escrita de código propriamente dito. O seu uso contribuiu, nas experiências de campo 'Programar é Fácil' da Universidade Júnior em 2015 e 2016, para uma desmistificação da programação, principalmente em idades entre os 12

[1] http://www.opensource.org/

e os 13 anos, onde os alunos não possuíam nenhuma experiência de programação, quer curricular, quer extracurricular.

Após a compreensão e exploração dos guiões com Scratch [ASO16], os alunos questionavam, diariamente, sobre de que modo é que a programação em Scratch se relacionava como uma linguagem de programação profissional. Este era o momento em que se fazia a ponte para o Python.

3.4.2 Python

A escolha sobre a principal linguagem de programação para este projeto foi o Python, por aproximar os jovens de uma ferramenta real de programação, algo bastante requerido pelo público-alvo, e por vários estudos apontarem para esta como uma linguagem com algum sucesso na iniciação à programação nesta faixa etária (dos 12 aos 18 anos de idade) [Geo08] [EEG+05] [GP13].

> "A linguagem na qual expressamos as nossas ideias tem uma forte influência no nosso processo de pensamento"
>
> — Donald Knuth, Literate Programming

Adicionalmente, o Python possui funcionalidades que contribuem imensamente para o processo de aprendizagem, como é o caso de criar animações simples e com elas criar-se os próprios jogos. Uma destas funcionalidades é o módulo *turtle* (tartaruga), uma maneira popular para introduzir a programação às crianças. Esta fazia parte da linguagem de programação original (LOGO) desenvolvida por Wally Feurzig e Seymour Papert em 1966 [2]. Outra funcionalidade é o módulo *tkinter* que é uma interface para o *Tk GUI toolkit* que permite, de um modo simples, criar programas com gráficos e animações um pouco mais avançados, com a vantagem de estar disponível na grande maioria das distribuições *Unix* [Lho07] e em sistemas *Windows* [Ric99]. Uma funcionalidade mais recente é a possibilidade da integração com o jogo Minecraft Pi Edition, com módulos que já estão incluídos na distribuição do sistema operativo escolhido para o Raspberry Pi.

3.4.3 Minecraft Pi Edition

Minecraft é um jogo de mundo aberto originalmente criado pelo designer sueco Markus "Notch"Persson, e mais tarde desenvolvido e publicado pela Mojang, adquirido pela Microsoft em Setembro de 2014 por 2.5 mil milhões de dólares americanos.

Os aspetos criativos e de construção do Minecraft permitem que os jogadores criem construções a partir de cubos com texturas e num mundo 3D processualmente gerado [Moj17].

Outras atividades no jogo incluem exploração, recolha de recursos, elaboração dos próprios elementos e combate. Vários modos de jogo estão disponíveis, incluindo um modo de sobrevivência onde o jogador deve adquirir recursos para construir o mundo e manter a saúde, um modo criativo onde os jogadores têm recursos ilimitados para construir e a capacidade de voar, um modo de aventura onde os jogadores podem jogar mapas personalizados criados por outros jogadores, e

[2] https://docs.python.org/2/library/turtle.html

um modo espetador onde os jogadores podem voar em redor e através de blocos, mas não pode colocar ou destruir nenhum [ZWCLJ13].

Na sua vertente de educação permite melhorar: o empenho dos alunos, ao trazer a aprendizagem para um meio em que já se encontram confortáveis; a colaboração, ao permitir que os alunos a trabalhem em grupo, para resolver problemas; a exploração criativa, ao dar-lhes a liberdade para explorarem um mundo aberto e desafiando-os a uma abordagem menos passo-a-passo e mais próxima do mundo real (como é o caso da tentativa-erro até atingirem o resultado pretendido) e os objetivos de aprendizagem tangíveis, ao criar salas de aula completamente inclusivas, desafiando os educadores a desenvolverem atividades para todos os tipos de alunos numa abordagem de aprender fazendo [Min16]. O Minecraft é um jogo excecional com muitas características que o tornam num ambiente atraente para a aprendizagem baseada em jogos. Este encoraja a resolução de problemas e a criatividade, e é imersivo e envolvente.

Este jogo é bastante popular entre os jovens, pelo que o seu uso em combinação com módulos de Python foi explorado na solução de oferta educativa desta dissertação. A ideia é que o uso de um ambiente já familiar para os alunos, como meio de ensino à programação, pode motivar e fomentar uma postura mais ativa por parte dos alunos, facilitando a aprendizagem.

Figura 3.2: Logótipo Minecraft Pi Edition

3.4.4 Trinket

Fundado em 2013 por 2 amigos, um engenheiro de software (Brian Marks) e um professor universitário de engenharia de software (Elliott Hauser), o Trinket permite ao utilizador escrever e executar código em Python a partir de um dispositivo móvel pessoal, desde que este possua um navegador de Internet relativamente recente (Firefox, Google Chrome, Internet Explorer ou Safari) e acesso à Internet, tudo isto sem ser preciso instalação, ou configuração, de qualquer outro tipo de software [Tri15]. É ainda possibilitado aos utilizadores que partilhem o seu código via e-mail, Twitter, Google+, Facebook e SMS através de um determinado link ou embebendo num outro web-site.

Figura 3.3: Logótipo Trinket

Existem algumas limitações ao usar este serviço, nomeadamente caso fique indisponível (*off-line*), situação em que também o código que lá está ficará inacessível, e o facto de o Trinket não suportar todas as bibliotecas Python [MR16].

Mais recentemente foi introduzida a capacidade de emular o Sense HAT (Secção 3.5.2) o que beneficiará bastante os jovens que pretendam experimentar este hardware, mas que ainda não o tenham adquirido (Ver Figura 3.4).

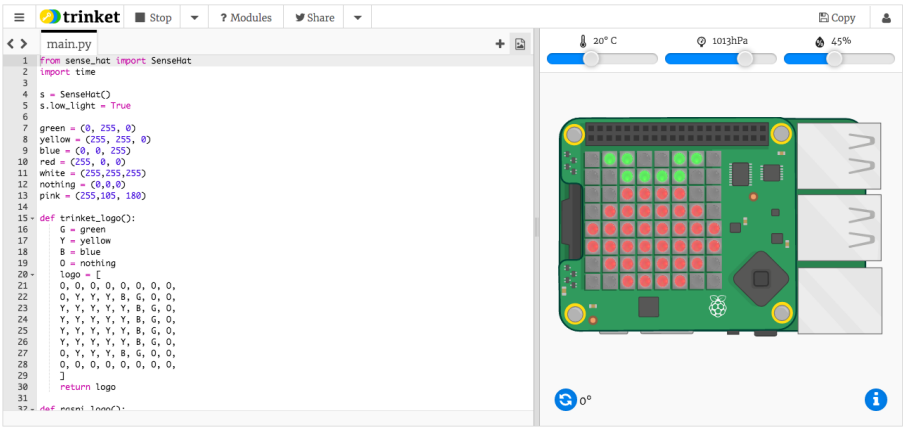


Figura 3.4: Trinket - Emulador do Sense HAT

3.4.5 GitHub educação

Como referido anteriormente (Secção 2.2.8) o GitHub educação é um segmento do popular GitHub destinado a estudantes e educadores. Uma vantagem clara da inclusão deste sistema na oferta educativa a desenvolver é evitar, quase por completo, a perda de um trabalho inteiro, pois o código de cada projeto está sempre disponível para todos os colegas do projeto e no próprio GitHub, onde é possível pesquisar pelo histórico do código e voltar para uma versão anterior, caso haja algum problema na versão atual.

O GitHub oferece ainda ferramentas para a gestão de projetos, como a gestão de problemas, discussões sobre código, entre outras, permitindo assim que o grupo de alunos esteja organizado e com objetivos claramente traçados em mente. O GitHub pode ser uma poderosa ferramenta de gestão de aprendizagem e, embora ofereça muitos benefícios, também há desafios envolvidos, ainda mais em idades como as que são focadas nesta dissertação. No entanto, quase todos os jovens em contacto com esta plataforma em atividades de campo foram capazes de usar com sucesso e conseguiram partilhar os seus projeto. Nesse sentido, as práticas sugeridas devem ser formadas e compartilhadas entre os educadores para melhor utilizar as capacidades do GitHub como uma plataforma de colaboração.

O GitHub foi também escolhido por permitir submissões de trabalhos e hospedar os conteúdo desenvolvidos, estes dois usos elementares são um reflexo da utilização atual dos professores em sistemas de gestão de aprendizagem típicos (por exemplo, Moodle e OpenEdx). A Figura 3.5 apresenta uma visão geral de alto nível das interações possíveis entre professores, alunos e conteúdos,

acrescentando depois as funcionalidades extra que o GitHub suporta nativamente em relação aos ambientes de aprendizagem tradicionais. Uma das principais diferenças, é a passagem de um papel passivo por parte do aluno, ao só ler conteúdo e passar a ter um papel em que contribui com material para a disciplina.

Figura 3.5: Os recursos adicionais que o GitHub fornece além dos recursos fornecidos pelos Sistemas de Gestão de Aprendizagem tradicionais (imagem traduzida de [ZFS+15])

3.4.6 GitBook

GitBook foi criado em meados de 2014 com a visão de criar uma solução moderna e simples para documentação, escrita digital e publicação. Este projeto começou por construir um formato de código aberto [3]. A filosofia é ser simples ao ponto de elegância, removendo distrações e preocupações dos criadores de conteúdo, para que eles possam escrever livremente. Desde então, tem crescido para ajudar mais de 250.000 pessoas a trabalhar juntas na redação de 150.000 livros que chegam a 20 milhões de visitantes a cada mês.

Oferece como principais funcionalidades a escrita em *Markdown* [4] ou *AsciiDoc* [5], geração de um livro digital no formato de um website ou *ebook* [6], multilingue, glossário, capa de livro, variáveis e *templates* [7], referências de conteúdo, *plugins* [8] e temas.

[3] https://github.com/GitbookIO/gitbook
[4] https://toolchain.gitbook.com/syntax/markdown.html
[5] https://toolchain.gitbook.com/syntax/asciidoc.html
[6] https://toolchain.gitbook.com/ebook.html
[7] http://toolchain.gitbook.com/templating/
[8] https://toolchain.gitbook.com/plugins/

O livro pode ser publicado de modo público, em que é completamente possível de ser pesquisado online e gratuito, ou privado onde apenas utilizadores autorizados têm acesso, mas esta trata-se de uma funcionalidade paga.

Figura 3.6: Logótipo GitBook

3.5 Hardware

Em termos de hardware, as escolhas recaíram sobre opções de muito baixo custo, com uma base de utilizadores grandes e em crescimento, de modo a evitar entraves monetários para serem incluídos em escolas, agrupamentos ou em ações individuais (*workshops*). As escolhas de hardware são descritas e discutidas nas próximas secções.

3.5.1 Raspberry Pi

Tendo em conta preço apelativo do Raspberry Pi (cerca de quarenta euros), e a versatilidade de projetos possíveis de ser implementados, este foi escolhido como alternativa ao uso de computadores convencionais. O seu uso permite, também, uma descentralização do *feedback* visual no monitor, para o qual os estudantes já estão "demasiado formatados", direcionando-os para *feedback* visual em componentes de *hardware*, como por exemplo: *LEDs* (Figura 3.7), motores, câmaras, etc.

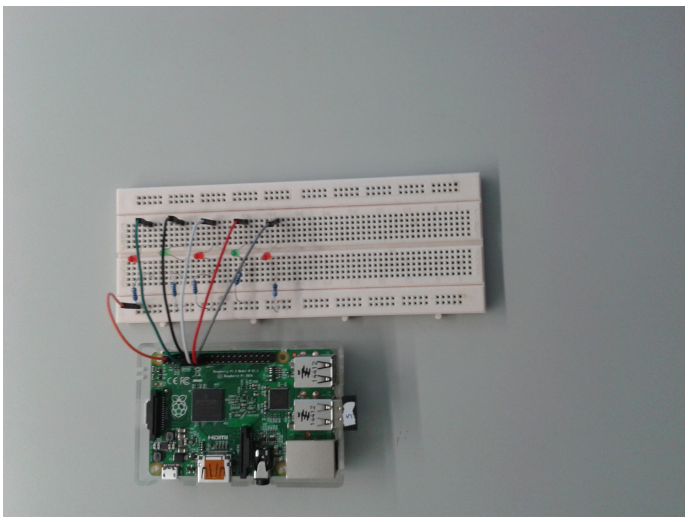


Figura 3.7: Raspberry Pi - Projeto com LEDs

Tendo em consideração a natureza iterativa em que a oferta educativa desta dissertação foi desenvolvida, o *feedback* dos testes de campo realizados, foi usado para decidir qual a versão do Raspberry Pi que deveria ser usada. Neste contexto, o *Raspberry Pi 3* é apontado como o candidato ideal para elaboração dos projetos de programação. Esta versão é mais apelativa em termos de caraterísticas (Tabela 3.1), permitindo eliminar episódios de "lentidão"causados pela falta de recursos das versões anteriores (Raspberry Pi 1 e Raspberry Pi 2) que levavam a momentos de frustração entre os jovens, e que não se verificam com o uso do Raspberry Pi 3.

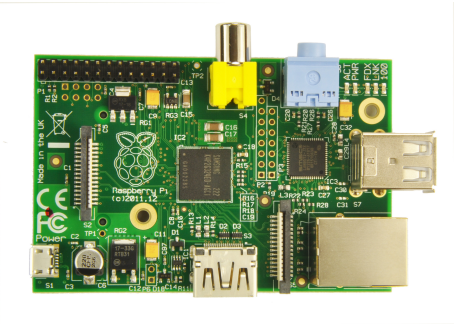


Figura 3.8: Raspberry Pi B

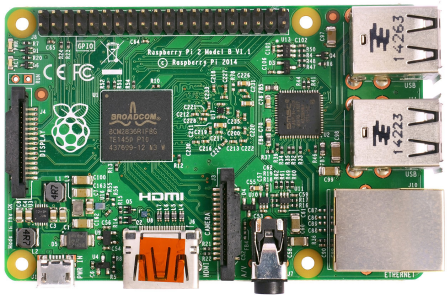


Figura 3.9: Raspberry Pi 2

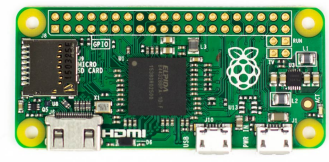


Figura 3.10: Raspberry Pi Zero

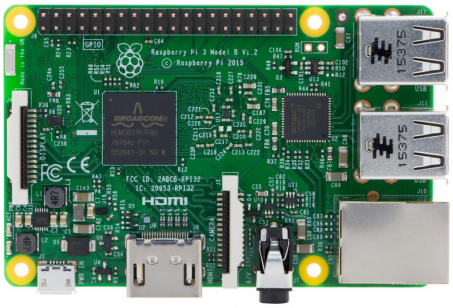


Figura 3.11: Raspberry Pi 3

Tabela 3.1: Comparação das principais versões do Raspberry Pi

Nome	Raspberry Pi B	Raspberry Pi 2	Raspberry Pi Zero	Raspberry Pi 3
Data de lançamento	Fevereiro 2012	Fevereiro 2015	Fevereiro 2015	Fevereiro 2016
CPU	ARM1176JZF-S	Cortex-A7	ARM1176JZF-S	Cortex-A53 64-bit
Frequência CPU	700MHz	900MHz	1GHz	1.2GHz
Nº de núcleos CPU	1	4	1	4
Memória RAM	512 MB	1 GB	512 MB	1 GB
Preço de lançamento	35€	35€	5€	35€

Como sistema operativo optou-se pelo Raspbian [9] por ser livre, baseado no Debian e otimizado para o hardware Raspberry Pi. O Raspbian foi criado de um modo independente, por Mike Thompson e Peter Green, fãs do hardware Raspberry Pi, dos objetivos educacionais da Fundação Raspberry Pi e, claro, do projeto Debian, mas é agora distribuído oficialmente pela Raspberry Pi Foundation. Um sistema operativo é o conjunto de programas básicos e utilitários que fazem o Raspberry Pi correr as suas aplicações. No entanto, o Raspbian fornece mais do que um sistema operativo puro, ele vem com mais de 35.000 pacotes, software pré-compilado e empacotado num formato para fácil instalação no Raspberry Pi. Ainda está em desenvolvimento ativo com ênfase em melhorar a estabilidade e o desempenho de tantos pacotes Debian quanto possível. Esta distribuição do sistema operativo para o Raspberry Pi conta já com uma grande comunidade global estendida a vários fóruns de discussão [10].

3.5.2 Sense HAT

O Sense HAT é uma placa adicional para o Raspberry Pi, feita especialmente para a missão Astro Pi lançada para a Estação Espacial Internacional em Dezembro de 2015.

As principais características do Sense HAT (Figura 3.12) são a inclusão de uma matriz de 64 *LEDs* RGB para mostrar gráficos e mensagens, um sensor de humidade que mede a humidade do ar, um sensor de pressão para medir a pressão do ar, um joystick que pode ser usado para cima,

[9] https://www.raspbian.org/
[10] https://www.raspberrypi.org/forums/viewforum.php?f=66

baixo, esquerda, direita e clicado e uma unidade de medição de inércia que possui: um acelerómetro que mede a força da aceleração, um giroscópio que mede a orientação e um magnetómetro que mede o campo magnético da Terra. Também foi criada uma biblioteca de Python que fornece um acesso facilitado a tudo na placa [20115].

Devido a possibilitar disponibilizar sensores já montados, sem necessidade de intervenção eletrónica, esta peça torna-se bastante importante para o projeto ao permitir ter um *feedback* fora do monitor, mas sem ocupar muito tempo com a parte eletrónica do projeto, permitindo aos jovens focarem-se na solução em código.

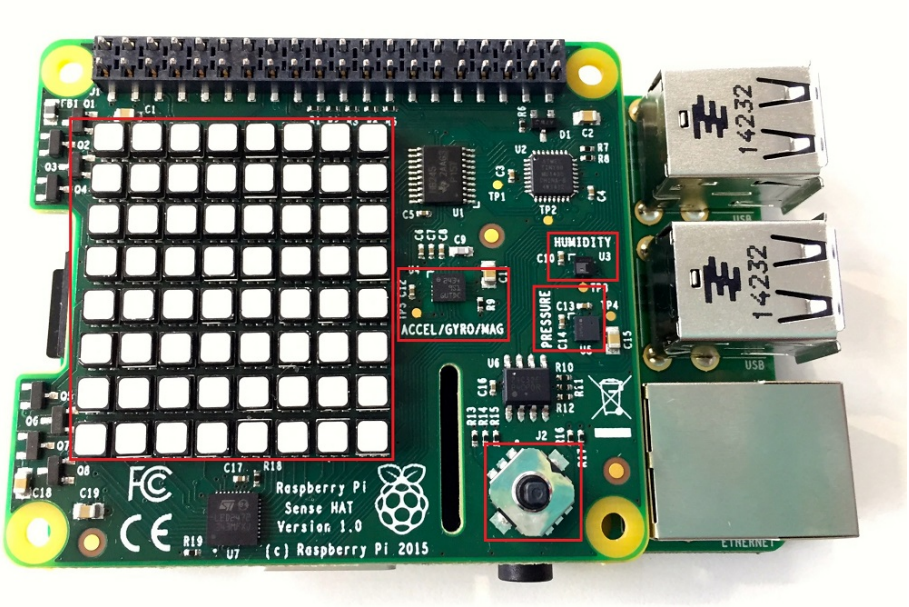


Figura 3.12: Sense HAT acopulado ao Raspberry Pi

3.5.3 Material eletrónico

Em alternativa ao Sense HAT, outro material foi escolhido em algumas atividades com idades mais jovens (12 aos 14 anos) devido ao impacto positivo na compreensão de como funciona um circuito elétrico, de o poderem manipular com código no Raspberry Pi, visualizarem esse efeito na mesa de trabalho e poderem recolher informação no Raspberry Pi com base em sensores no seu circuito (Ver Figura 3.13). A ideia é esta abordagem poder permitir uma mais fácil desmistificação do uso da programação, ao relacioná-la com circuitos elétricos externos ao computador.

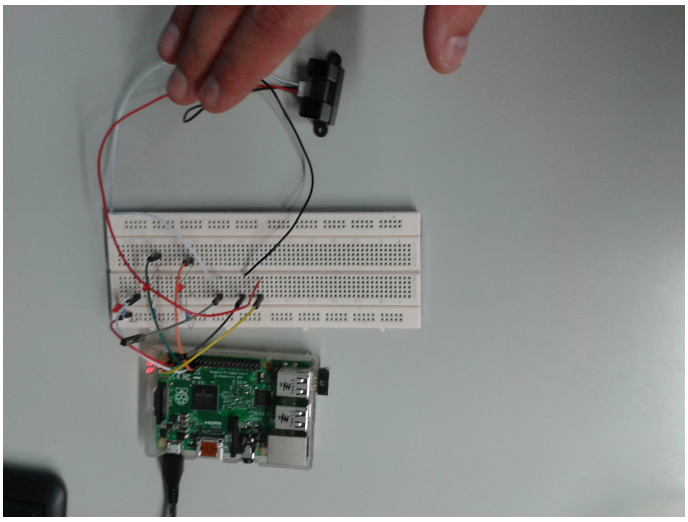


Figura 3.13: Raspberry Pi - Projeto com LEDs e um sensor de proximidade

Globalmente escolheram-se *LEDs*, resistências, botões, sensores de distância, sensores de proximidade e câmaras web.

3.6 Resumo

O ensino da programação distingue-se das demais áreas do conhecimento por não se enquadrar num modelo formal de ensino, nas idades do público-alvo. Os conhecimentos de programação devem ser transmitidos num enquadramento de raciocínio lógico associado à resolução de problemas e desenvolvimento de soluções tecnológicas de um modo estruturado e colaborativo, como é necessário em ciência da computação.

Como tal, a conceção de uma oferta educativa especificamente desenhada e concebida para a transmissão destes conhecimentos foi abordada neste capítulo, destacando-se as várias componentes que definem a solução: software, hardware e estratégias pedagógicas a seguir, e a criação de unidades didáticas exemplo que podem ser usadas para ensaios de campo e iterações da solução. As escolhas foram definidas com o foco na conceção de uma solução que relacione programação com o mundo físico fora do computador (mecanismo de *feedback* físico) e outras ciências, e que aproxima a programação de projetos DII (*Do It Yourself*, faz tu mesmo). Para alunos e professores isto traduz-se em mais diversão, independência e escolha e num ambiente de apoio e colaboração para aprender novas habilidades.

Capítulo 4

Conteúdos e atividades desenvolvidas

Motivados pela ideia que não há uma solução universal para todos, e tendo em conta os principais objetivos desta dissertação, desenvolveram-se 4 atividades que se distinguem essencialmente em termos de público-alvo a atingir e estratégias pedagógicas selecionadas. Estas atividades foram avaliadas em ensaios de campo concebidos com o intuito de recolher opinião dos principais intervenientes nas atividades, que foi usada para melhorar a proposta.

Este capítulo está subdividido em 4 secções que detalham os objetivos, material e métodos de ensino usados em cada atividade, assim como a discussão dos resultados dos ensaios de campo.

4.1 Programar é Fácil

A atividade Programar é Fácil da Universidade Júnior, destinada a um público-alvo dos 12 aos 14 anos de idade, foi renovada em conteúdos e métodos pedagógicos de edições anteriores. Carateriza-se por um conjunto de pequenos projetos cuja realização não ultrapassa os 90 minutos, e baseia-se num método de ensino mais apoiado. Esta atividade tem como objetivo geral introduzir e sensibilizar os jovens para programação, desmistificando a programação e o medo de estragar o computador.

4.1.1 Objetivos

Os principais objetivos desta atividade são:

- Compreender como funciona um computador e identificar os seus componentes.
- Criar pequenos blocos de código capazes de resolver problemas bem definidos e fechados.
- Compreender a interface entre programação e mundo físico fora do computador.
- Compreender e controlar, com programação, circuitos elétricos simples.
- Entender os conceitos de abstração, decomposição e automação.

4.1.2 Material

Esta atividade foi desenhada de modo a contemplar o seguinte hardware: Raspberry Pis, LEDs, resistências, sensores (distância e movimento), botões, câmaras web, breadboards e fios de ligação.

O software a utilizar é: Scratch e Python, com o ambiente de desenvolvimento IDLE 3.

Foram reformulados os primeiros nove guiões e criados outros cinco. Estas atividades apesar de isoladas, encontram-se sequencialmente ligadas entre si, que podem ser consultadas em maior detalhe no *website* da atividade *Programar é Fácil!* [1]:

1. Raspberry Pi – O que é? Como funciona? - Onde são demonstrados e explicados os vários componentes que integram o Raspberry Pi.

2. Raspberry Pi – Como montar o RPi? - Onde é demonstrado como se montam os vários componentes de *hardware* do Raspberry Pi.

3. Raspberry Pi – Como instalar o Sistema Operativo? - Onde é demonstrado como se instalam os diversos sistemas operativos disponíveis para o Raspberry Pi.

4. Raspberry Pi – Como aceder ao RPi? - Onde se demonstra como aceder remotamente ao Raspberry Pi usando *VNC*.

5. Raspberry Pi – Como configurar o VNC? - Onde se demonstra como configurar o *VNC* no Raspberry Pi de modo a refletir o ecrã que é transmitido via *HDMI*.

6. Raspberry Pi – Vamos experimentar o RPi! - Onde se demonstra a configuração de um pequeno circuito com um LED ligado ao Raspberry Pi.

7. Raspberry Pi – Vamos Programar o RPi! (Figura 4.1)- Onde se detalha a elaboração de um projeto usando *Scratch* que aborda alguns conceitos básicos de programação.

8. Raspberry Pi – Projeto Jogo de Luzes? (Figura 4.2) - Onde se relaciona a montagem de um circuito com vários LEDs aliados à programação com *Scratch* para os animar e tonrá-los num jogo.

9. Raspberry Pi – Projeto Alarme - Onde se detalha a elaboração de um projeto com recurso a um sensor de luz e a um laser para criar um alarme com recurso a *Scratch*.

10. Raspberry Pi – Programar em Python - Onde se demonstra um tutorial de fácil compreensão para iniciação na linguagem de programação *Python*

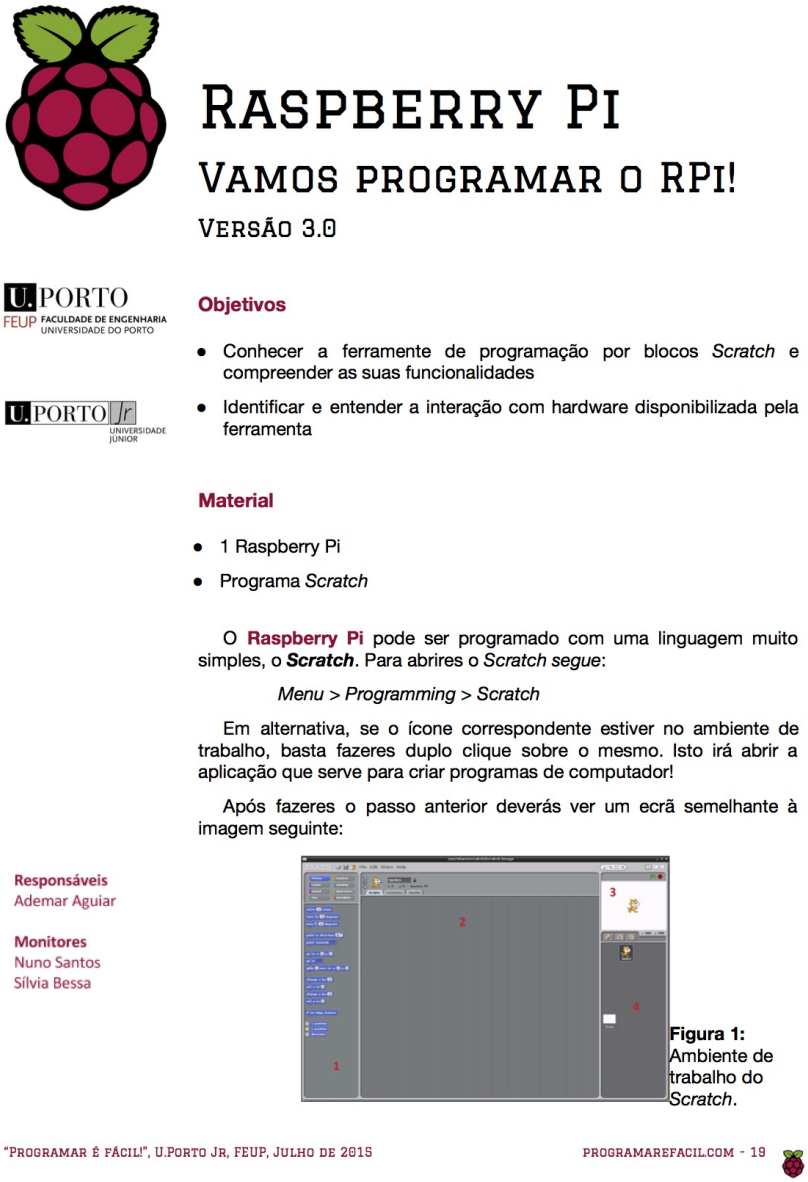
11. Raspberry Pi – Stop Motion - Onde se demonstra e exemplifica um projeto que permite a realização de um filme em *Stop Motion* com recurso a uma câmara web.

12. Raspberry Pi – Media Center - Onde se demonstra como criar o nosso próprio media center com o Recurso a um Rasberry Pi.

[1] https://github.com/Coding4Kids/programarefacil/wiki

13. Google Cardboard – Realidade Virtual - Onde os alunos experimentam vários projetos já feitos em realidade virtual com um telemóvel e um *Google Cardboard*.

14. Roteiros de Exploração Livre (Figura 4.3) – Jogos - Onde se dá a possibilidade de escolherem várias experiências alternativas de jogos com recurso a programação.



RASPBERRY PI

VAMOS PROGRAMAR O RPI!

VERSÃO 3.0

U.PORTO FEUP FACULDADE DE ENGENHARIA UNIVERSIDADE DO PORTO

U.PORTO JR UNIVERSIDADE JÚNIOR

Objetivos

- Conhecer a ferramente de programação por blocos *Scratch* e compreender as suas funcionalidades
- Identificar e entender a interação com hardware disponibilizada pela ferramenta

Material

- 1 Raspberry Pi
- Programa *Scratch*

O **Raspberry Pi** pode ser programado com uma linguagem muito simples, o ***Scratch***. Para abrires o *Scratch segue*:

Menu > Programming > Scratch

Em alternativa, se o ícone correspondente estiver no ambiente de trabalho, basta fazeres duplo clique sobre o mesmo. Isto irá abrir a aplicação que serve para criar programas de computador!

Após fazeres o passo anterior deverás ver um ecrã semelhante à imagem seguinte:

Responsáveis
Ademar Aguiar

Monitores
Nuno Santos
Sílvia Bessa

Figura 1: Ambiente de trabalho do *Scratch*.

"PROGRAMAR É FÁCIL!", U.PORTO JR, FEUP, JULHO DE 2015 PROGRAMAREFACIL.COM - 19

Figura 4.1: Guião Programar é Fácil - Vamos programar o RPi!

Figura 4.2: Atividades com Scratch

Figura 4.3: Atividades no Code.org

Foi ainda disponibilizada uma página para a iniciativa onde todos os participantes ou curiosos pela atividade podem descarregar livremente o conteúdo elaborado, ou até sugerirem alterações

ou esclarecimentos. Podem ainda contactar com os responsáveis da atividade e recolher mais informações sobre as atividades futuras que se encontram de momento a ser planeadas.

4.1.3 Métodos de Ensino

Esta atividade segue uma aprendizagem baseada em projetos de pequena duração, bastante controlados e com passos minuciosamente detalhados. Depende bastante do uso do método pedagógico demonstrativo, que consiste na transmissão de técnicas visando a repetição do procedimento através da demonstração: explicação - demonstração - aplicação. É um método que promove uma aprendizagem rápida, onde os objetivos remetem para uma execução correta da tarefa em questão.

Numa fase inicial apresenta-se uma visão global do que os alunos irão aprender, onde se dá uma explicação geral sobre o conteúdo, como se irá processar a tarefa e o tempo que é expectável ocuparem com o desenvolvimento da mesma, a um ritmo normal. De seguida inicia-se a explicação passo a passo, que é feita a um ritmo propositadamente mais lento, de modo a todos alunos poderem acompanhar e executar em simultâneo. Caso se justifique, por exemplo em tarefas mais complexas), poderá ainda seguir-se uma terceira etapa onde se resume sinteticamente o processo de resolução da tarefa.

Os alunos são convidados a trabalharem em pares ao longo do dia, de modo a discutirem, programarem e montarem os seus circuitos sempre em grupo. Existe um espaço aberto para dúvidas durante o decorrer de toda a sessão.

4.1.4 Discussão dos resultados

Foi elaborada uma proposta à Universidade do Porto no âmbito da Universidade Júnior, de modo a ser realizada uma atividade de Verão para jovens do sétimo, oitavo e nono ano com o intuito de os introduzir à programação, chamada *Programar é Fácil!*[2]. Esta foi considerada uma oportunidade única de aferir os conhecimentos de programação de uma parte do público-alvo (dos 12 aos 14 anos de idade) a que se destina parte do trabalho desta dissertação. Assim, durante um dia inteiro e quatro semanas de atividades, foram recebidos 16 alunos diferentes por dia, que realizaram várias atividades diretamente relacionadas com programação. Esta atividade reforçou a ideia inicial de que havia necessidade de se desenvolver um programa de ensino de programação de qualidade, dadas as normais lacunas de conhecimento nos alunos. Nas duas últimas edições desta atividade (2015 e 2016) que ocorreram com a minha colaboração, estima-se que tenham contactado com a atividade desenvolvida, e tendo em conta pontuais ausências, aproximadamente 580 alunos (280 em 2015 e 300 em 2016), do sétimo ao nono ano de escolaridade, sendo que o trabalho desta dissertação recai sobre a edição de 2016.

No decorrer da atividade foi recolhido algum *feedback* informal dos alunos relativamente às competências de programação que possuem. Foi aferido também se já tinham contactado com programação na escola, seja curricularmente ou em algum programa extracurricular, os ritmos de

[2] https://universidadejunior.up.pt/atividades.php?a=programar-e-facil

aprendizagem que preferiam, ou que estão habituados a contactar, as atividades que lhes apelavam mais e o que os frustrava mais no decorrer das atividades. Esta informação foi preciosa para o ajuste adequado dos guiões, atividade e plataformas desenvolvidas.

No Programar é Fácil, foram testadas experiências alternativas com Minecraft (Figura 4.4) e a linguagem de programação Python, com base nos guiões ingleses da *Raspberry Pi Foundation*, explicados pelos monitores da atividade. A motivação dos alunos para trabalharem com o Minecraft foi desde logo observável, todos os alunos queriam estar a "jogar", quando na verdade, estavam a programar em Python sem se aperceberem. Existe uma clara vantagem em trazer o ensino da programação para um ambiente em que lhes é familiar e onde se divertem: os alunos ficam mais motivados, tendo-se constatado, ainda que com base nas opiniões recolhidas pelos monitores, que o ritmo de aprendizagem era superior.

Figura 4.4: Primeiro teste com Minecraft

Devido às idades do público-alvo desta atividade (12 a 14 anos de idade), não existem hábitos de exploração independente, estando sempre dependentes do monitor da atividade para avançar, pondo em pausa qualquer atividade assim que se apresente uma dúvida. É normal e esperado que existam bastantes dúvidas nesta faixa etária por muitos dos jovens estarem pela primeira vez a contactar com programação e tratando-se de uma atividade 1 dia apenas, o que traz algumas exigências na hora de elaborar conteúdo para ser percebido por este grupo. Foram vários monitores que acompanharam os alunos e que foram uma motivação na hora de idealizar guiões que fossem de fácil compreensão para os alunos e completos o suficientemente para os monitores (alunos da Faculdade de Engenharia). Estes guiões, estavam minuciosamente detalhados nos passos a seguir de modo a evitar confusões e permitir alguma autonomia aos jovens. Os guiões eram

complementados com demonstrações, com uma explicação a ritmo normal, muito resumida, e a segunda mais lenta e passo a passo. Caso a primeira não seja resumida ou em forma de introdução, geram-se de imediato dúvidas, pois os jovens vão tentar executar de imediato, o que não abona o ritmo de uma turma inteira.

Após este tempo a melhor realização destas experiências continua a ser um momento de genuíno espanto dos jovens, quando programam um LED e ele acende nas suas secretárias, pois continuam a esperar que o *feedback* do que escreveram no computador apareça no ecrã e não no mundo físico, no seu campo de visão.

4.2 Hackathon: Cidade + Inteligente

A atividade Hackathon: Cidade + Inteligente é destinada a um público-alvo entre os 15 e 18 anos de idade. Carateriza-se por ser uma atividade de maior duração (uma semana), com aprendizagem baseada em projetos e que apela a uma maior autonomia dos jovens. Esta atividade tem como objetivo geral integrar a programação com metodologias ágeis de desenvolvimento de software e motivar os jovens para a engenharia informática numa solução (protótipo) que beneficie a cidade e/ou a vida dos jovens presentes na atividade.

4.2.1 Objetivos

Os principais objetivos desta atividade são:

- Conceber, decompor e desenvolver projetos DYI.
- Reconhecer projetos similares que permitam adaptar soluções para o projeto.
- Compreender *Scrum* como método ágil de desenvolvimento de software.
- Comunicar assertivamente e colaborar com os colegas para a resolução de problemas.
- Identificar erros de implementação.
- Reconhecer a utilidade de sítios on-line para esclarecimento de dúvidas e fonte de soluções para erros comuns de implementação.
- Entender as fases de conceção e desenvolvimento de software.

4.2.2 Material

Esta atividade foi desenhada de modo a contemplar o seguinte hardware: Raspberry Pis, Sense HATs e LEGO Mindstorms.

O software selecionado foi: Python, com o ambiente de desenvolvimento IDLE 3 e MySQL.

A atividade é ainda acompanhada por wiki desenvolvida para apoio da atividade [3], e um repositório de partida, e contempla ainda algum material extra, nomeadamente Post-its, para o quadro físico de gestão de tarefas.

4.2.3 Métodos de Ensino

Como sugerido pelo nome da atividade, o Hackathon: Cidade + Inteligente tem como inspiração os concursos de desenvolvimento de ideias tecnológicas. A atividade está essencialmente concebida para uma semana de desenvolvimento de projeto, embora possa ser adaptada para as salas de aula considerando-se que um *sprint* equivale ao período disponível de tempo de uma aula. Em termos categóricos, a atividade segue uma metodologia de PBL, na qual os alunos são convidados a apresentar soluções para uma problemática previamente definida, tendo que as implementar seguindo métodos ágeis de desenvolvimento de software e programação em pares, num período de implementação que culmina com a criação de uma demo e uma apresentação pública.

A atividade inicia-se com um pequeno diálogo de aferição sobre os conhecimentos de programação dos seus participantes e as suas motivações.

De seguida, e após apresentar o propósito da atividade aos alunos, segue-se uma sessão de *brainstorming*, onde individualmente cada aluno vai conceptualizando o que acha que são os problemas da sua cidade ou do seu dia a dia, e escreve cada uma dessas ideias num Post-it. Esses Post-its são depois agregados em grandes categorias - *clustering* - (Figura 4.5) como Lazer, Domótica, Cidade, entre outros.

Figura 4.5: Agrupamento das ideias iniciais em categorias gerais

Após esta categorização, convidam-se os alunos a dirigirem-se às categorias que mais os motivam e tentam convencer outros 3 colegas a juntarem-se, de modo a formarem um grupo (Figura 4.6).

[3] https://github.com/coding4kids/cidadeinteligente/wiki/Recursos

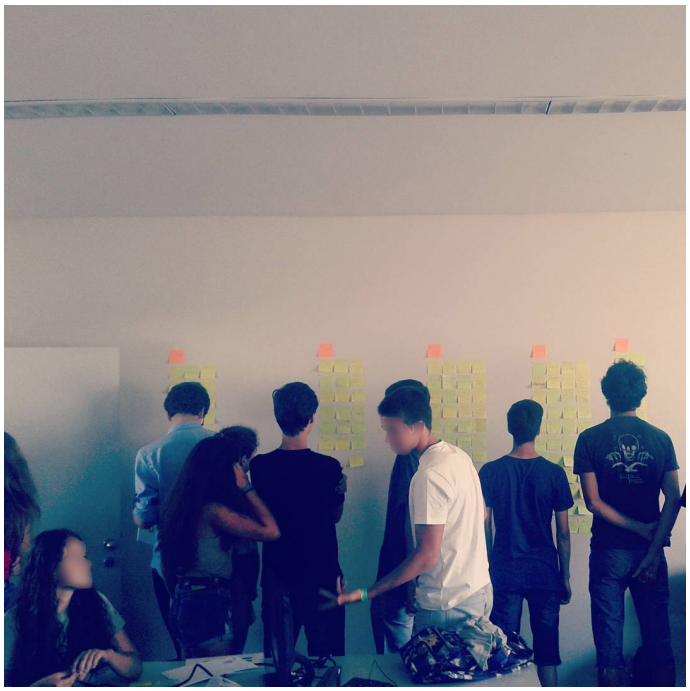


Figura 4.6: Formação dos grupos

Neste momento, após a formação dos grupos, é criada uma interrupção propositada onde se introduzem os guiões de introdução ao Raspberry Pi e programação do Sense HAT com Python, para que os jovens tenham uma noção real das capacidades do Raspberry Pi e da panóplia de funcionalidades oferecidas pelo Sense HAT.

Antes de se iniciar o projeto segue-se, ainda, um pequeno tutorial de como usar o motor de busca Google para o esclarecimento de dúvidas de programação. Estimula-se, assim, a independência dos grupos, que é fundamental dada a enorme variedade de projetos e a impossibilidade dos monitores da atividade estarem em permanência com todos os grupos. Além disso, existe já à partida uma coleção de recursos na WIKI da atividade pronta para esclarecer algumas dúvidas comuns. Posteriormente, introduz-se o método de como colocar dúvidas no desenrolar da atividade (Figura 4.7). Resumidamente após, pesquisarem no Google e na WIKI da atividade, se a dúvida persistir, o jovem é convidado a expor em voz alta a sua dúvida, com o intuito de estimular a estruturação do problema com que se deparam. Caso a dúvida já tenha sido esclarecida anteriormente, convida-se o colega a quem tenha sido prestado o esclarecimento a explicar agora pelas suas próprias palavras, incentivando um ambiente de colaboração intergrupo.

Agora que os jovens, já sabem pesquisar, questionar e o que o hardware e software são capazes de fazer, inicia-se o processo de refinamento da ideia do projeto. Aqui, e em grupo, pretende-se que os jovens acordem entre si quais devem ser as principais funcionalidades da sua solução. Estas são escritas num Post-it e coladas no respetivo quadro de gestão do projeto da equipa (*Kanban board*). Este quadro divide-se em 3 grandes categorias: para fazer (*to do*), a fazer de momento (*doing*) e pronto (*done*).

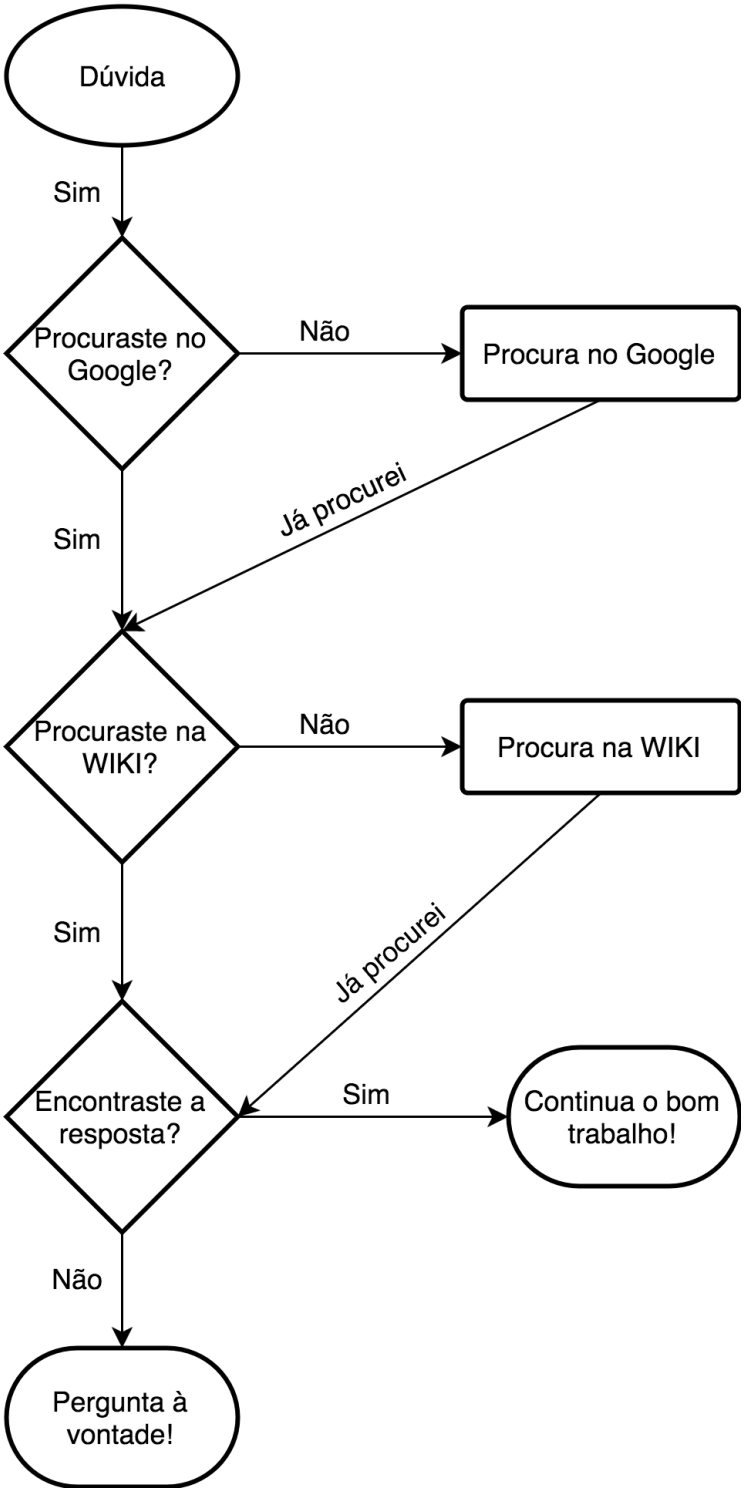


Figura 4.7: Diagrama de fluxo para colocar uma dúvida

É nesta altura que se introduz o *Scrum*, que pode então relacionar-se com o ensino do seguinte modo: O proprietário do produto (o monitor) sugere uma lista de funcionalidades para o produto, chamado de*backlog*, priorizados por valor para o produto final. Cada item tem critérios de aceitação, que podem ser definidos, ou não, pelo professor e é similar a um plano de aula, mas trabalhado a partir do produto final para trás. O *backlog* do produto não é algo estático, o professor deve modificá-lo conforme o projeto progride e com base no *feedback* dos alunos.

O próximo passo é o planeamento do Sprint e para tal é necessário um *Scrum* Master, que é um dos alunos do próprio grupo, capaz de manter o grupo focado nos seus objetivos e eliminar problemas na execução do projeto. No primeiro *sprint* o monitor poderá ter que assumir este papel também.

Durante o planeamento do *sprint*, o grupo, o *Scrum Master* (aluno do grupo) e o monitor (se necessário) dividem as funcionalidades do *backlog* em tarefas e decidem o que podem completar no *sprint* atual, tendo em conta os recursos disponíveis e fazer um *sprint backlog* (lista de tarefas).

No caso do ensaio de campo foram usados *sprints* de meio dia (manhã e tarde), com *stand up meetings* (reuniões em pé) no início de cada um. Os elementos do grupo levantam-se, dizem uns aos outros o que conseguiram alcançar, o que eles vão fazer de seguida, discutem quaisquer questões, desafios e ajudas necessárias. Foram usados grupos de 4 elementos, é aconselhado um mínimo de 3 elementos para que a atividade corra normalmente.

No final de cada *sprint*, o trabalho deve estar pronto a usar. Se não, ele é enviado de volta para o *backlog* e pode ser reintroduzido no próximo *sprint*, se necessário.

Regra geral o *sprint* termina com uma revisão do mesmo, em que a equipa de desenvolvimento e o proprietário do produto inspecionam o que fizeram até ao momento e pensam sobre o que ainda precisa ser feito. No entanto, com os jovens a quem se destina foi necessário um refinamento desta retrospetiva, devido aos *sprints* de curta duração, tendo sido deslocada para o início de cada *stand up meeting*. Nesta restroespetiva é analisado se os critérios de aceitação são cumpridos (por exemplo: Esta função de medir a temperatura, está a recolher a temperatura ambiente como é suposto ou está a gerar valores aleatórios de teste apenas?). Este é o ponto onde o monitor pode aceitar ou rejeitar o trabalho se ele não cumprir os critérios acordados e, acima de tudo, sugerir como é que o grupo pode melhorar no próximo *sprint*.

Para iniciar o próximo *sprint*, o grupo escolhe mais funcionalidades do *backlog* do produto, subdivide-o em tarefas, faz o seu planeamento de *sprint* e começa novamente a trabalhar.

No penúltimo dia da semana da atividade os grupos fazem as suas demonstrações de treino (Figura 4.8), onde mostram slides e vídeos das suas soluções, recebem *feedback* dos outros grupos sobre o que melhorar, o que não ficou claro na apresentação e tentam melhorar até ao dia seguinte.

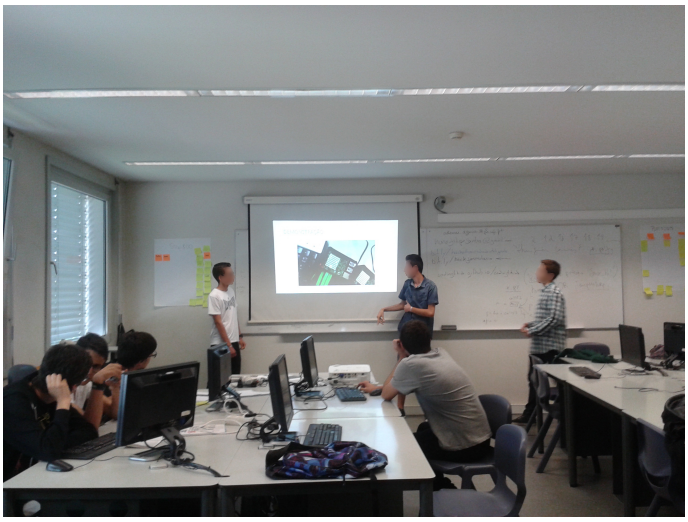


Figura 4.8: Apresentação intermédia do projeto Despertador Inteligente

No último dia da atividade os grupos fazem os ajustes finais no seu produto para a apresentação (Figura 4.9 e iniciam o processo de publicação online no GitHub, onde fazem um clone de um repositório de exemplo da atividade, acrescentam os detalhes dos seus projetos e fazem um *merge* no repositório original de modo a submeter o seu projeto. Este tipo de publicação é estimulado como forma de promover a partilha de código entre grupos, que fica disponível para consulta após as apresentações de cada grupo.

Figura 4.9: Apresentação final do projeto House Keeper

4.2.4 Discussão dos resultados

Foi elaborada uma proposta à Universidade do Porto, em 2016, também no âmbito da Universidade Júnior, de modo a ser realizado um projeto semanal de Verão para jovens do nono,

décimo e décimo primeiro ano com o objetivo principal de desmistificar a programação com uma série de desafio tecnológicos, chamada *Hackathon: Cidade + Inteligente*[4]. Esta foi uma atividade de elevado valor ao permitir definir a direção, o interesse e o tipo de abordagens que mais se adequam à faixa dos jovens pré-universitários para esta dissertação. Durante uma semana estes jovens formaram grupos, desenvolveram ideias sobre como melhorar a sua cidade, colaboraram e desenvolveram capacidades de resolução de problemas com recurso a programação. Desta atividade resultaram 13 projetos publicamente disponíveis na página do GitHub da atividade[5], com vídeos de demonstração e instruções para cada projeto. Esta atividade contou com a participação de sensivelmente 60 alunos ao longo das 4 semanas. Com esta atividade de 5 dias pretendia-se motivar os alunos para a engenharia informática ensinando todo o ciclo de vida de uma aplicação, desde os seus requisitos de utilização real, até à sua programação concreta, instalação final e observação da sua utilização. As aplicações a construir pelos participantes, preferencialmente propostas pelos participantes, formadores e parceiros, tiveram como objetivo fundamental melhorar algum aspeto da sua cidade, tornando-a mais inteligente, possivelmente envolvendo aquisição de dados, atuação, monitorização e gestão de dados, mobilidade, segurança e ambiente, entre outras ideias e áreas de aplicação. Os projetos foram baseados nos Raspberry Pi, nos seus sensores e periféricos, e desenvolvidos utilizando Python.

Foi aferido inicialmente, assim como na atividade anterior, se os alunos já tinham contactado com programação na escola, ou extra-curricularmente, de onde apenas 12.5% dos alunos tinham efetivamente contactado com programação. Dos que tinham programado, Python, Pascal e Scratch foram as principais tecnologias usadas. Esta informação foi uma vez mais determinante para a escolha de tecnologias a usar.

Destacou-se ainda o facto de os alunos preferirem ferramentas de programação menos infantis como é o caso do Scratch, por falhar, na opinião deles, a transporem os conceitos abordados na altura de usarem ferramentas mais sérias. Certo é que esta opinião dos alunos convergiu com outra opinião de um investigador dedicado a esta problemática[6]. Como a motivação e divertimento dos alunos é fundamental para a interiorização de conhecimento [AK00], optou-se pelo uso de Python como linguagem base da atividade, sendo que existem resultados bastante satisfatórios com uso desta linguagem para a introdução a programação [Geo08].

Foram também adotadas metodologias ágeis para o desenvolvimento das suas aplicações no decorrer da semana, com *Scrum Meetings*, *Sprint Reviews* e um simples *Kanban Board* com *Post-its* (*Todo, Doing, Done*), onde cada *Post-it* correspondia a uma tarefa. As tarefas eram estimadas em *T-Shirt Size* (1-Fácil,2-Médio,3-Difícil), e atribuídas a cada par dentro do grupo, o que se revelou suficiente para a compreensão dos conceitos.

Cada elemento do par programava alternadamente em intervalos de 45 minutos com uma pausa de 5 minutos entre cada intervalo. Ajudando deste modo a que ambos estivessem a par da tarefa

[4]https://universidadejunior.up.pt/atividades.php?a=hackathon-cidade-inteligente
[5]https://github.com/Coding4Kids/cidadeinteligente/tree/master/Projetos
[6]https://www.edsurge.com/news/2013-05-28-opinion-learning-to-code-isn-t-enough

que se encontravam a desenvolver, evitando que um elemento mais proficiente se apodere do desenvolvimento, ou que um colega mais passivo fique mais em segundo plano.

Cada *sprint* correspondia a um meio dia e globalmente no final do segundo dia da atividade os alunos já conseguiam entender a sua velocidade do *sprint* e estimar corretamente quantas tarefas conseguiam concluir em cada *sprint*. Haviam ainda tarefas surpresa durante o decorrer da atividade, propostas pelos monitores, de modo a desafiar os alunos a resolverem problemas mais complexos por si e a levarem as suas aplicações a um nível superior de desenvolvimento, tendo grande parte conseguido alcançar um protótipo com uma interface gráfica de utilizador.

Tinham ainda 2 apresentações com demonstrações das suas soluções, uma no quarto dia, de modo a aferirem o estado real do seu projeto e receberem *feedback* dos colegas, e outra no último dia da atividade. No final, faziam um *fork* do repositório GitHub da atividade, organizavam o seu projeto enviavam para os seus repositórios pessoais e pediam para fazer um *merge* no repositório do projeto.

Algo importante testado também com este grupo foi o estímulo à pesquisa, em que se fornecia uma literatura organizada para consulta, na página da atividade, e um tutorial de pesquisa de problemas de programação no Google. Caso estas instâncias falhassem a esclarecer as questões dos alunos, pedia-se que estes formulassem as suas dúvidas oralmente o que muitas vezes os ajudava a pensar melhor nas suas dúvidas e acabando por esclarece-las sem necessitarem de intervenção. Por muitas dúvidas serem recorrentes, convidava-se sempre os colegas de outros grupos a esclarecer as mesmas.

Durante esta atividade, foi pedido aos participantes que preenchessem um questionário para recolha de *feedback*, sendo este anónimo e facultativo. Este questionário incluía questões sobre o gosto pela atividade, a sua relevância para o percurso académico dos jovens, o conteúdo (respostas de 1 a 5), e questões de categorização nomeadamente sobre a vontade de ingressar no ensino superior e escolher a área das engenharias (respostas Sim/Não). Foi ainda colocada uma questão de resposta aberta onde os jovens podiam referir aspetos que achavam que deviam ser melhorados (Anexo A). Assim, além das opiniões que muitos alunos decidiram transmitir diretamente aos monitores da atividade, foram obtidas respostas de 36.7% dos alunos, num total de 22 inquéritos válidos.

Os resultados mostram que a atividade (Figura 4.10), conteúdo (Figura 4.10) e a sua relevância (Figura 4.10) foram sempre avaliados de modo positivo, sendo que a maioria dos alunos os classificaram com a nota máxima.

Dos jovens inquiridos, 95% pretendiam ingressar no Ensino Superior (Figura 4.14), sendo que desses, 86% consideravam escolher cursos relacionados com as áreas de engenharia (Figura 4.13).

Relativamente aos aspetos a melhores, 86% dos inquiridos não tinham nada apontar. 9% dos inquiridos referiram aspetos de software, nomeadamente substituir o ambiente de desenvolvimento de software (IDLE3) por uma alternativa visualmente mais apelativa. Os restantes 5% correspondem à resposta de um jovem que indicou que um aspeto a melhorar seria dar mais apoio à programação. No entanto, é interessante ressalvar que dentre os inquiridos, este foi o único

jovem que não frequentava a área de Ciências e Tecnologias no ensino secundário. Além destas respostas, um dos inquiridos aproveitou o campo de resposta aberta para referir o seguinte:

> "Para mim a atividade foi fantástica, já que me permitiu perceber que realmente adoro programar e considero a vida de programadora como um dos caminhos que possa seguir. Agradeço a colaboração dos monitores que foram impecáveis e nos motivaram e ajudaram sem dizer exatamente o que tínhamos de fazer mas motivando-nos a pesquisar e aprender por nós próprios. Gostava muito de voltar e se puder ainda volto a passar por aí!"
>
> — Participante da atividade Hackathon: Cidade + Inteligente

Estes resultados confirmam o interesse dos jovens nos conteúdos e estratégia de ensino desenvolvidos, dando indicações positivas sobre a adequação da mesma ao público-alvo em questão. Referir, em particular, que a atividade foi escolhida essencialmente por jovens a frequentar a área de Ciências e Tecnologias, com interesse em seguir o ensino superior, nomeadamente áreas de engenharia, o que valida o uso de estratégias pedagógicas que mais do que ensinar programação, a integram com metodologias de desenvolvimento de software.

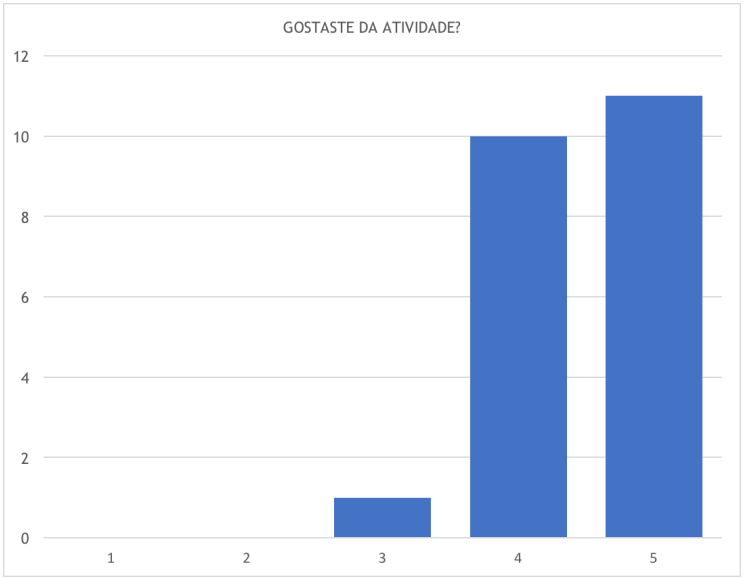


Figura 4.10: Resultados relativamente ao gosto pela atividade Hackathon: Cidade + Inteligente

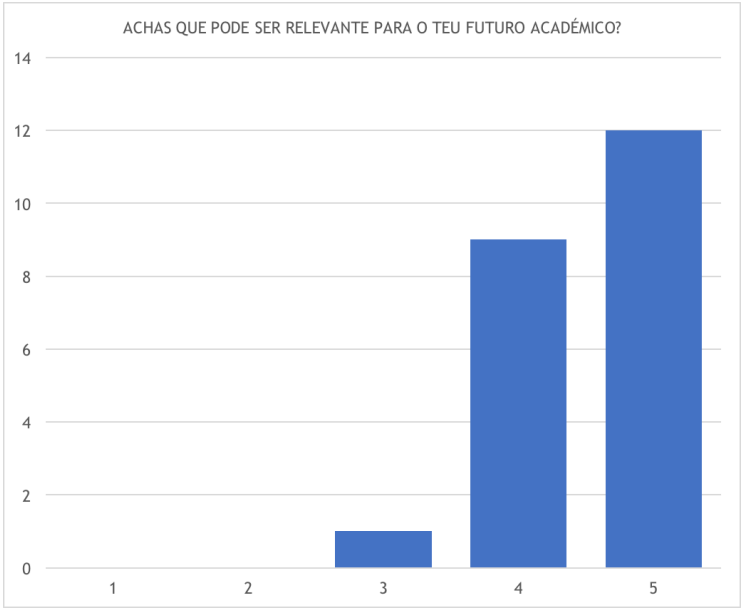


Figura 4.11: Resultados relativamente à relevância atividade Hackathon: Cidade + Inteligente

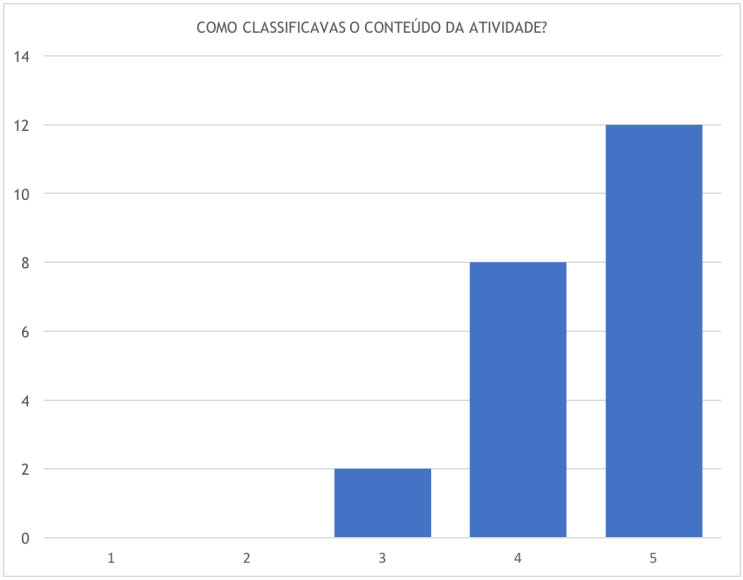


Figura 4.12: Resultados relativamente à adequação dos conteúdos atividade Hackathon: Cidade + Inteligente

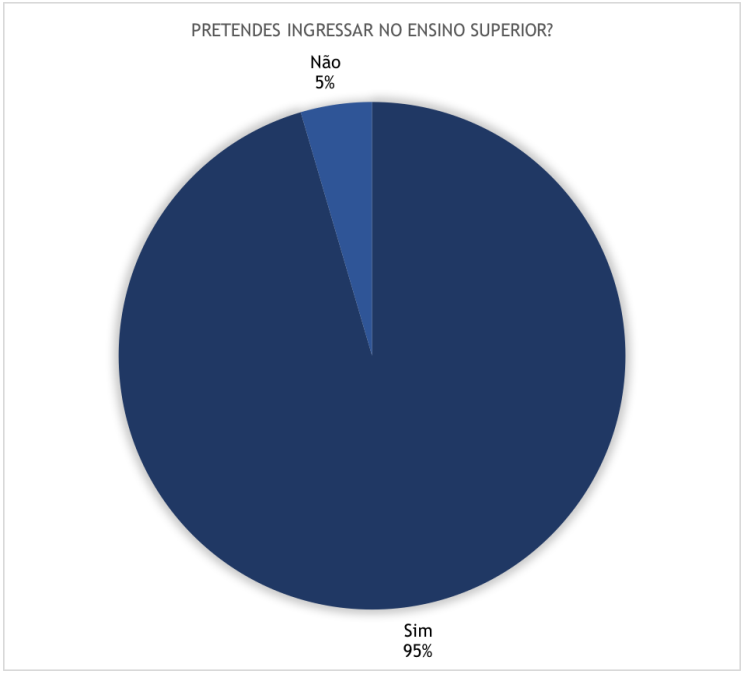


Figura 4.13: Distribuição dos alunos que frequentaram a atividade Hackathon: Cidade + Inteligente e pretendem ingressar no ensino superior

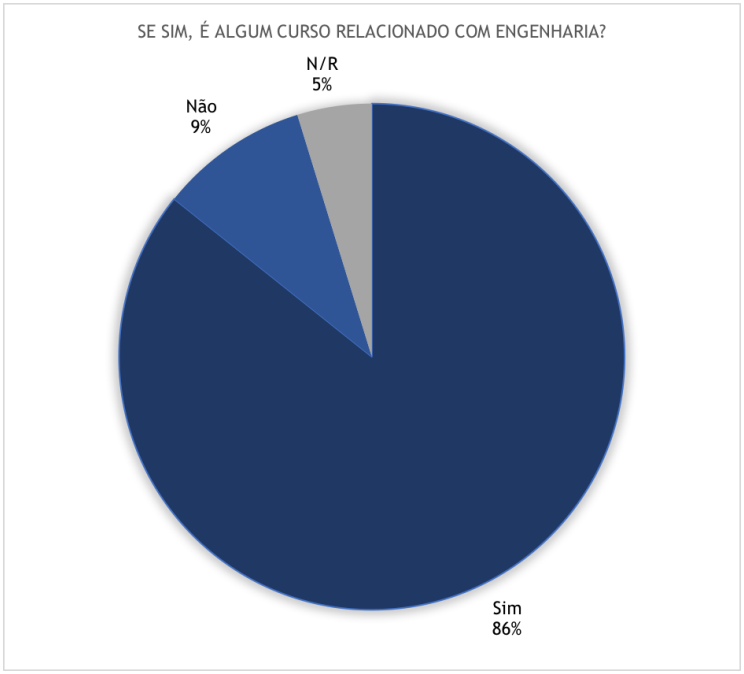


Figura 4.14: Distribuição dos alunos que frequentaram a atividade Hackathon: Cidade + Inteligente e consideram ingressar em cursos na área das engenharias

4.3 Programação para Jovens

A atividade Programação para Jovens foi idealizada para a um público alvo dos 12 aos 18 anos de idade, sendo o resultado da fusão das duas atividades anteriores. Figurando como uma oferta educativa agregadora de conteúdos que pode ser adaptada a diferentes tipos conhecimento e ritmos de aprendizagem dos jovens que nela participem. Carateriza-se por ser dinâmica, apelativa e promover a autonomia dos jovens, enquanto transmite as várias áreas do conhecimento associadas à programação e desenvolvimento de software, sem as adulterar. Pode ser usada em ambiente formal de aula, ou sobre a forma de atividade extracurricular ou workshop, sendo capaz agrupar no mesmo ambiente jovens de diferentes faixas etárias sem que seja comprometida a aprendizagem de cada um dos seus intervenientes. Esta atividade partilha os objetivos gerais desta dissertação, sensibilizar, promover e facilitar o ensino da programação, podendo ser interpretada como a solução integradora que se propôs desenvolver, embora não contemple todas as especificidades das duas atividades anteriores.

4.3.1 Objetivos

Os principais objetivos desta atividade são:

- Entender os conceitos de abstração, decomposição, automação, pensamento algorítmico, generalização e identificação de padrões;
- Compreender o conceito de depuração e testes;
- Compreender a interface entre programação e mundo físico fora do computador;
- Compreender e controlar, com programação, circuitos elétricos simples;
- Reconhecer projetos similares que permitam adaptar soluções para o projeto.
- Comunicar assertivamente e colaborar com os colegas para a resolução de problemas.
- Reconhecer a utilidade de sítios on-line para esclarecimento de dúvidas e fonte de soluções para erros comuns de implementação.

4.3.2 Material

Esta atividade foi desenhada de modo a contemplar o uso do: Raspberry Pis, Sense HATs, LEDs, resistências, breadbords, fios de ligação, câmaras web, botões e sensores (distância e movimento). O software selecionado foi o Minecraft, Scratch, Python, com o ambiente de desenvolvimento IDLE 3, e o Trinket.

O GitBook foi selecionado como plataforma de distribuição do conteúdo em formato de livro digital. Este livro (Anexo B) é composto por 4 capítulos com um total de 6 unidades didáticas que respeitam a estrutura definida na Secção 3.2.1.

1. Ligar LEDs com o Raspberry Pi - Onde se monta um circuito simples e se programa com a Shell, Scratch e Python;

2. Jogo dos LEDs com Scratch - Onde se tenta criar um jogo de luzes com Scratch;

3. Começar com o Sense HAT - Onde se exploram todas as funcionalidades do Sense HAT com Python;

4. Começar com o Minecraft - Onde se demonstram todas as funcionalidades da biblioteca do Minecraft para Python;

5. Cabine de Fotografia no Minecraft - Onde se cria uma cabine que tira fotografias no mundo real com ações geradas no Minecraft;

6. O mapa do Minecraft com o Sense HAT - Onde se cria um mapa do mundo em redor do jogador do Minecraft e se aprendem conceitos avançados de programação;

Possui ainda um glossário (Anexo B.14) por existirem bastantes conceitos transversais ans atividades e uma página com uma coleção de recursos (Anexo B.13), que se desdobra em tutoriais, consulta, projetos, *software*, *hardware* e vídeos.

4.3.3 Métodos de Ensino

Os métodos de ensino nesta atividade dependem das faixas etárias em questão. Separando em 2 grupos, dos 12 aos 14 (A), e dos 15 aos 18 (B), define-se as seguintes estratégias:

Para o primeiro segmento (A) é um ensino PBL em conjunto com métodos pedagógicos demonstrativos pontuais para dúvidas comuns dos vários grupos. A programação em pares, apesar de encorajada não é obrigatória, mas observa-se um progresso mais lento em atividades com apenas um elemento face ao desenvolvimento das atividades em pares.

No segundo segmento (B) a abordagem é também PBL, com método pedagógico interrogativo com processo indutivo onde se conduzem os alunos a uma solução deduzida pelos próprios, em vez de fornecida pelo monitor. Dada a natureza curta destas atividades não se torna necessária a aplicação de programação em pares nesta faixa etária.

4.3.4 Discussão dos resultados

Esta atividade foi testada no 1º Movimento Código Portugal, em Lisboa, e na atividade 'Programação para Todos' do CSI Labs, realizada na Faculdade de Engenharia da Universidade do Porto (Figura 4.15), tendo os resultados sido bastante satisfatórios.

Figura 4.15: Atividade Programação para todos

Um fator distintivo desta atividade prende-se com jovens em anos de transição (14, 15 anos) que conseguem assim ter uma progressão mais natural, ao ser disponibilizados vários conteúdos de diferentes níveis de conhecimento de programação. O carácter exploratório, por design, desta solução não exige que todos os jovens tenham o mesmo nível de conhecimento e possam ir traçando os seus próprios percursos, ou sequencia de atividades.

Existe uma clara vantagem ao usar o Minecraft como uma das ferramentas para o ensino de programação, que não vem usualmente mencionada de uma forma clara na literatura, mas que é observável em todas as atividades onde foi usado: a diferença de postura dos jovens quando é explicado um conceito de programação numa ferramenta comum, ou no Minecraft. Regra geral os jovens tomam uma atitude menos passiva e mais inquisitiva sobre de que modo os conceitos de programação abordados influenciam a construção do projeto no Minecraft e conseguem muito facilmente começar a relacionar variáveis, com funções, com dicionários no Python, porque entendem a sua aplicabilidade no mundo do Minecraft.

Houve um enorme interesse no formato do livro digital apresentado com mais de 1300 visualizações desde a sua apresentação em Dezembro, que se dividem pelo globo como indicado na Figura 4.16.

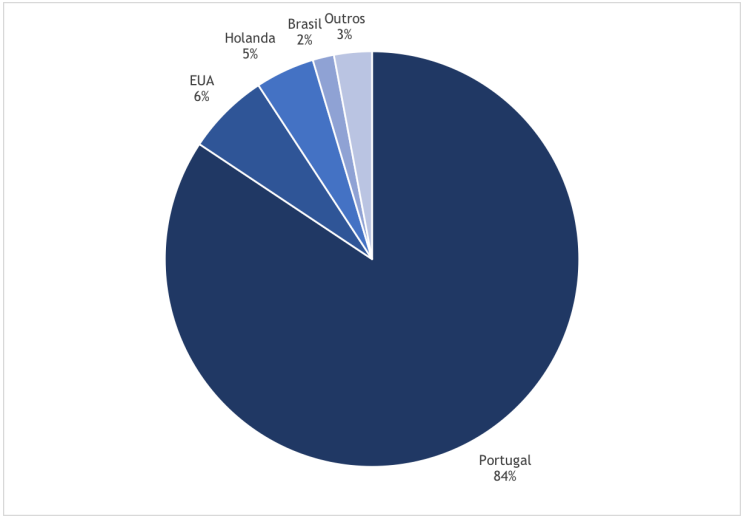


Figura 4.16: Número de visualizações por país do livro Programação para Jovens

Este livro foi configurado de modo a incluir vídeos com explicações, imagens com movimento (gifs), uma ferramenta para correr código Python e ainda emular o Sense HAT (Figura 4.17), este livro está em integração contínua com um repositório GitHub, a partir do qual também pode ser editado. Para que qualquer pessoa, mesmo que tecnicamente mais limitada, possa sugerir alterações ou esclarecimentos ao livro (Figura 4.18), bastando para isso que aproxime o cursor do paragrafo que pretende ver alterado, clicar no e comentar.

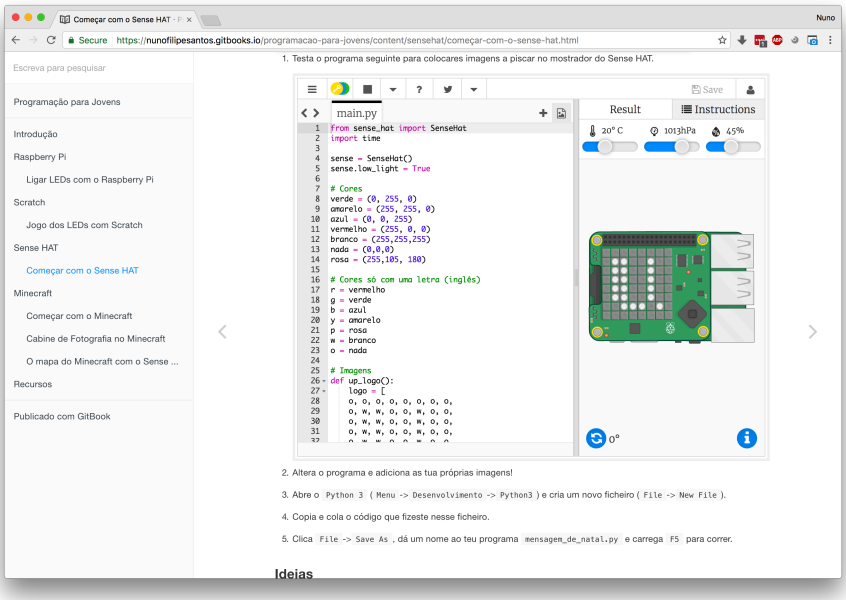


Figura 4.17: Funcionalidade de emulação do Sense HAT com o Trinket

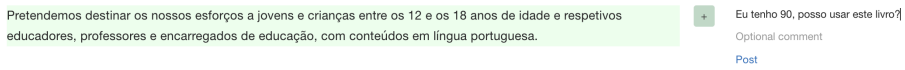


Figura 4.18: Funcionalidade de sugestão de alterações

Foi ainda ativa a funcionalidade de subscrição do livro que permite que sempre que seja publicado um novo capítulo os seus subscritores recebam uma notificação. Este livro é concebido com os educadores em mente, de modo a poder ser facilmente integrado de modo curricular ou extracurricular numa escola.

4.4 Dinamização do Ensino da Programação

Alternativamente às secções anteriores, que descrevem atividades destinadas aos jovens, a presente secção detalha uma proposta de intervenção pedagógica cujo público-alvo são professores de ensino básico e secundário de áreas tecnológicos.

O Ensino de Programação desde cedo nas escolas é uma das áreas com maior crescimento à escala mundial. Qualquer escola atualmente sente uma necessidade premente de se posicionar com o futuro no horizonte, quer para satisfazer os estudantes, quer para agilizar o trabalho dentro da própria escola. Muitas vezes, apenas uma curta e eficaz formação específica, com mão-de-obra especializada em Tecnologias de Informação e Comunicação, consegue-se motivar muito os alunos para a aprendizagem da programação recorrendo a planos com tecnologias muito atuais e de baixo custo.

Esta proposta de intervenção pedagógica propõe suprimir esta lacuna de conhecimento nos participantes, fornecendo instrumentos e noções práticas de novas tecnologias, garantindo assim um desenvolvimento acompanhado destas capacidades. Deste modo, esta formação munirá os seus formandos de competências que lhes possibilitarão desenvolver e dinamizar o ensino da programação e colaboração em projetos de software e hardware nas suas turmas. Está neste momento a aguardar aprovação pelo Conselho Científico-Pedagógico de Formação Contínua no Instituto Piaget de modo a poder ser devidamente creditada para as carreiras dos docentes, daí a seguir esta estrutura fixa que será detalhada de seguida.

4.4.1 Objetivos

Com esta formação pretende-se que os formandos:

- Reconheçam a importância do ensino da programação
- Reconheçam novas oportunidades de aprendizagem na área das novas tecnologias
- Reconheçam as várias plataformas disponíveis para dinamizar o ensino da programação

Pretende-se ainda que os formandos adquiram conhecimentos e capacidades em:

- Desenvolvimento de um plano de aula na área da programação
- Desenvolvimento de conhecimentos de manipulação de periféricos de hardware, em concreto Raspberry Pi
- Desenvolvimento de conhecimentos de ferramentas colaborativas

No final desta formação, os formandos deverão ser capazes de:

- Compreender os conceitos fundamentais associados ao ensino da programação
- Distinguir os modos de programação individual e em pares
- Compreender a importância da gestão de projetos e relacioná-la com a colaboração
- Conhecer as várias ferramentas de acesso livre disponíveis para o ensino de programação
- Instalar com sucesso as ferramentas de programação no Raspberry Pi
- Compreender a importância dos vários módulos de hardware disponíveis para Raspberry Pi
- Conceber, ao longo da formação, de um modo criativo e distintivo, um plano de aula com recurso a um projeto prático desenvolvido a partir das ferramentas abordadas na formação

4.4.2 Plano

Este foi o plano proposto para a atividade:

1. Introdução ao Ensino da Programação
   (a) Ensino da Programação
   (b) Algoritmia
   (c) Iniciativas Mundiais
   (d) Arranque do Projeto Global
2. Introdução às Ferramentas Colaborativas
   (a) Google Drive
   (b) GitHub
   (c) Open edX/Moodle
   (d) Primeiro checkpoint do Projeto Global
3. Introdução ao Raspberry Pi
   (a) O que é o Raspberry Pi
   (b) Software para o Raspberry Pi

   (c) Hardware para o Raspberry Pi

   (d) Segundo checkpoint do Projeto Global

4. Projetos de Programação em Grupo

   (a) Dinâmicas de colaboração

   (b) Exemplos de projetos didáticos

   (c) Ferramentas de apoio à programação

   (d) Desenvolvimento dos projetos globais

4.4.3 Metodologias

Estrutura da realização das sessões:

- Presencial inicial (8h)
- Semi-presencial: online, síncrono e a distância (13h)
- Presencial intermédio (8h)
- Semi-presencial: online, síncrono e a distância (12h)
- Presencial final e avaliação (9h)

Número de horas previstas por cada tipo de sessões:

- Sessões presenciais conjuntas: 25 horas
- Sessões semi-presenciais: 25 horas

4.4.4 Avaliação

Para este tipo de intervenção pedagógica é necessário o desenvolvimento de elementos de avaliação:

- Participação presencial e online
- Produtos resultantes da ação (disponibilizados no ambiente Moodle das disciplinas lecionadas)
- Auto-avaliação (relatório de reflexão crítica)

O que implica avaliação pelo formador, formandos e um avaliador externo, que afere a qualidade global da atividade.

4.4.5 Projeto Global

O projeto global terá como principal objetivo a criação de uma atividade de apoio ao Ensino de Programação para jovens. Como tal, os formandos deverão ter em mente o público alvo (idade, conhecimentos, atenção, nível de leitura e compreensão...), o que pretendem que seja passado aos alunos e um modo de os avaliar.

De modo a praticar a colaboração em projetos de código, poderão trabalhar com colegas de outras zonas do país onde esteja a decorrer a formação.

4.5 Resumo

Neste capítulo foram apresentadas as principais atividades e conteúdos desenvolvidos nesta dissertação, salientando-se a natureza iterativa do processo, que tira partido das informações recolhidas nos ensaios de campo para atualizar as atividades, conteúdos ou inserir novos conceitos em atividades subsequentes. Foram apresentadas 3 atividades destinadas a jovens, sumariadas na Tabela 4.1, assim como uma atividade de dinamização do ensino da programação, concebida como uma proposta de intervenção pedagógica, endereçada a professores do ensino básico e secundário.

Tabela 4.1: Resumo das atividades desenvolvidas para o ensino de programação a jovens

Atividade	Programar é Fácil	Hackathon: Cidade + inteligente	Programação para Jovens
Objetivo-geral	Introduzir e sensibilizar os jovens para programação, desmistificando a programação e o medo de estragar o computador	Integrar a programação com metodologias ágeis de desenvolvimento de software e motivar os jovens para a engenharia informática	Partilha os objetivos gerais desta dissertação: sensibilizar, promover e facilitar o ensino da programação
Público-alvo	Jovens dos 12 aos 14	Jovens dos 15 aos 18	Jovens 12 aos 18
Material	RPIs, LEDs, resistências, sensores (distância e movimento), botões, câmaras web, breadboards e fios de ligação, Scratch, Python e Guiões de atividades.	RPIs, SenseHATs, LEGO Mindstorms, Python e MySQL (opcional), GitHub, wiki da atividade	RPIs, Sense Hats, LEDs, resistências, breadbords, fios de ligação, câmaras web, botões e sensores Scratch, Python , Trinket, Minecraft
Metodologia	Ensino Apoiado, PBL com projetos de pequena duração	PBL,*Scrum* e Programação em pares	Ensino apoiado ou autónomo, PBL, programação pares (opcional)
Ensaio de Campo	Oficinas de Versão, UJr 2016	Verão em Projeto, UJr 2016	Movimento Código Portugal e Programação para todos

Os resultados sugerem a adequação da estratégia e conteúdos desenvolvidos face aos objetivos propostos, evidenciando que o ensino da programação baseado em projetos aliado a ferramentas que apelem à criatividade dos jovens, tornam o ensino da programação mais apelativo.

Capítulo 5

Conclusões

Este capítulo final apresenta uma revisão da informação, e experiência, desta dissertação, assim como uma exposição da investigação e futuro do mesmo.

5.1 Satisfação dos Objetivos

O principal objetivo desta dissertação visava o desenvolvimento de um conjunto de unidades didáticas (conteúdos e atividades) para o ensino de programação para jovens com idades compreendidas entre os 12 e os 18 anos, englobando também os seus educadores e escolas.

A ideia passava por causar um impacto positivo nesta área de ensino, que tirasse proveito das iniciativas mundiais a decorrer na área.

De modo a alcançar este objetivo, todas as ideias inicialmente propostas, nomeadamente a criação de agentes de transmissão de conhecimento, como planos, conteúdos e seleção de ferramentas, foram desenvolvidas com sucesso. Foram desenvolvidos e libertados de forma completamente gratuita e aberta os resultados desta dissertação, que foram além dos testes de campo inicialmente propostos. Isto permitiu um refinamento dos conteúdos desenvolvidos, tendo ainda conseguido-se observar a clara motivação dos alunos, e inclusive encarregados de educação, perante a abordagem de *project based learning* selecionada no âmbito desta dissertação.

Finalmente, foi bastante satisfatório que o trabalho desenvolvido no âmbito desta dissertação tenha sido já convidado para alguns programas piloto em Portugal, estando-se neste momento a trabalhar para aprovação de um programa de dinamização tecnológica para professores do ensino secundário, cujo o objetivo é criar uma oferta devidamente creditada para a carreira de docente.

5.2 Trabalho Futuro

Apesar dos resultados obtidos preencherem os objetivos inicialmente propostos, o trabalho desenvolvido tem um caráter exploratório e de prova de conceito que deve continuar a ser desen-

volvido no futuro, por exemplo em termos de conteúdos, ou mecanismos de recolha contínua de *feedback* dos intervenientes.

A nível dos conteúdos, tendo sido desenvolvidos vários planos de aula para o ensino de programação para crianças, estes serão melhorados e atualizados em cada edição dos futuros workshops no CSI Labs, assim como nas escolas piloto parceiras desta iniciativa já em 2017. Além do mais, serão desenvolvidos videoaulas e tutoriais de modo a levar estas atividades de ensino de programação a novos alunos e interessados, pelo canal do CSI Labs no YouTube[1], de modo a complementar e enriquecer o material já existente.

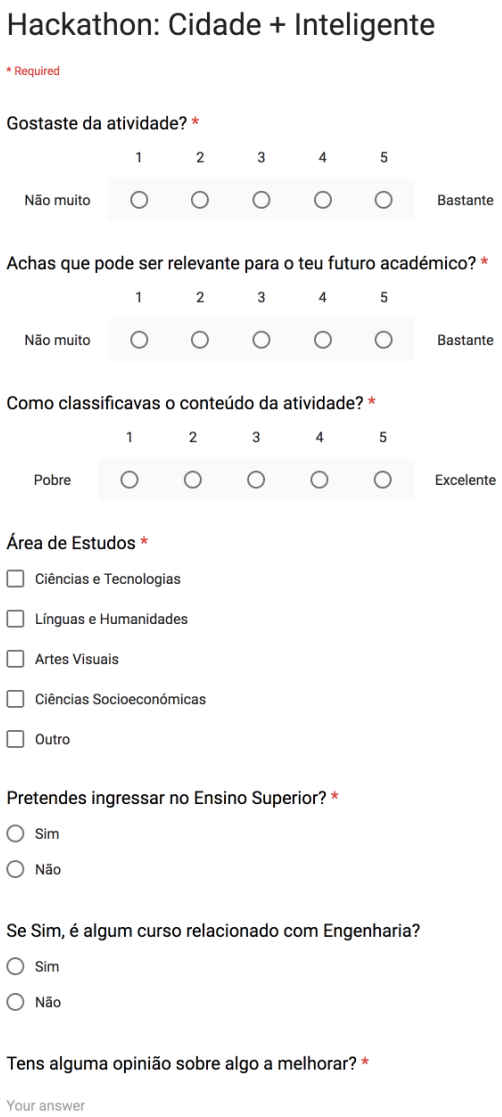
A nível da recolha de *feedback* por parte dos participantes, procurar-se-á um método menos intrusivo do que um questionário e que automatize a recolha de informação, em tempo real, permitindo assim o foco na procura de uma métrica de sucesso a definir no futuro.

Finalmente, pretende-se ainda avaliar outros tipos de conhecimento lecionados no ensino superior em cursos de engenharia informática, além dos já validados, que possam vir a mostrar-se relevantes para a compreensão da introdução à programação, tendo em conta sempre a motivação dos alunos.

[1] https://www.youtube.com/channel/UCoQ8DVG8pfqC6-s4naVan7w

Anexo A

Formulário de *feedback* Hackathon: Cidade + Inteligente


Hackathon: Cidade + Inteligente

* Required

Gostaste da atividade? *

	1	2	3	4	5	
Não muito	○	○	○	○	○	Bastante

Achas que pode ser relevante para o teu futuro académico? *

	1	2	3	4	5	
Não muito	○	○	○	○	○	Bastante

Como classificavas o conteúdo da atividade? *

	1	2	3	4	5	
Pobre	○	○	○	○	○	Excelente

Área de Estudos *

- ☐ Ciências e Tecnologias
- ☐ Línguas e Humanidades
- ☐ Artes Visuais
- ☐ Ciências Socioeconómicas
- ☐ Outro

Pretendes ingressar no Ensino Superior? *

- ○ Sim
- ○ Não

Se Sim, é algum curso relacionado com Engenharia?

- ○ Sim
- ○ Não

Tens alguma opinião sobre algo a melhorar? *

Your answer

Figura A.1: Formulário de *feedback* da atividade Hackathon: Cidade + Inteligente.

Anexo B

Programação para Jovens

Figura B.1: Capa do livro Programação para Jovens.

B.1 Introdução

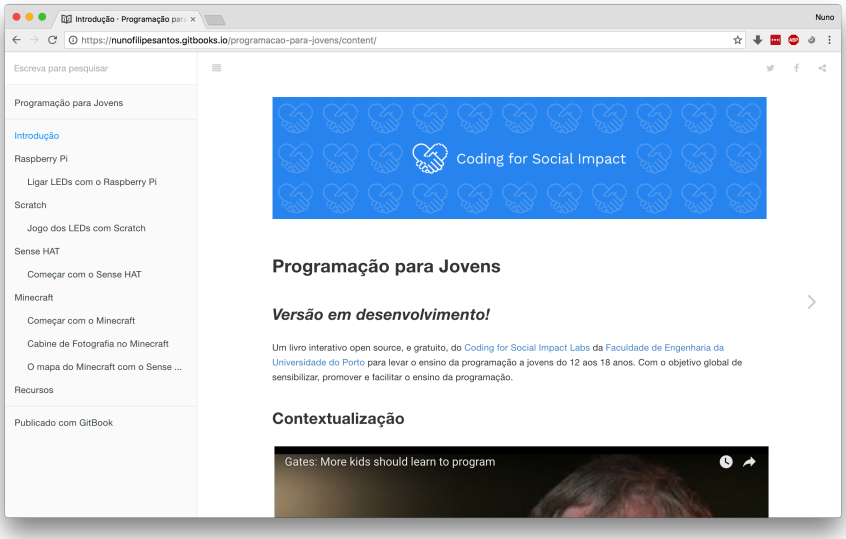


Figura B.2: Introdução.

B.2 Raspberry Pi

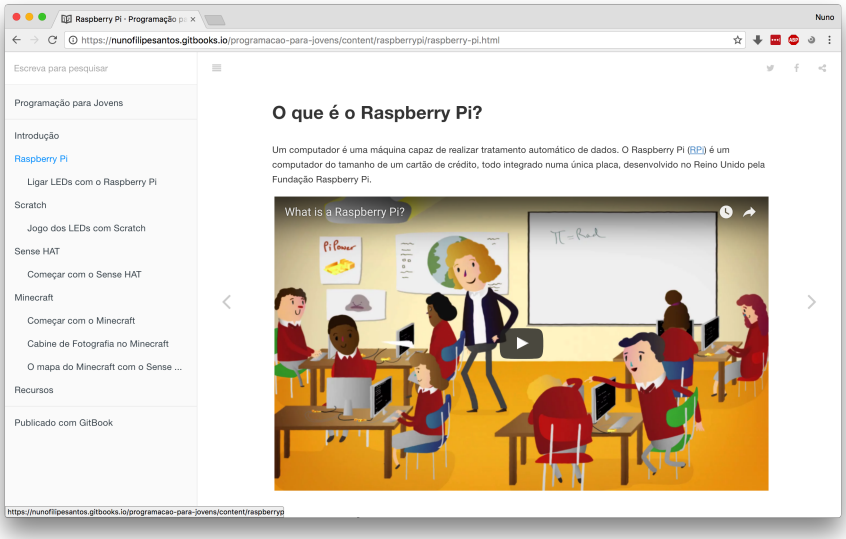


Figura B.3: Capítulo: Raspberry Pi.

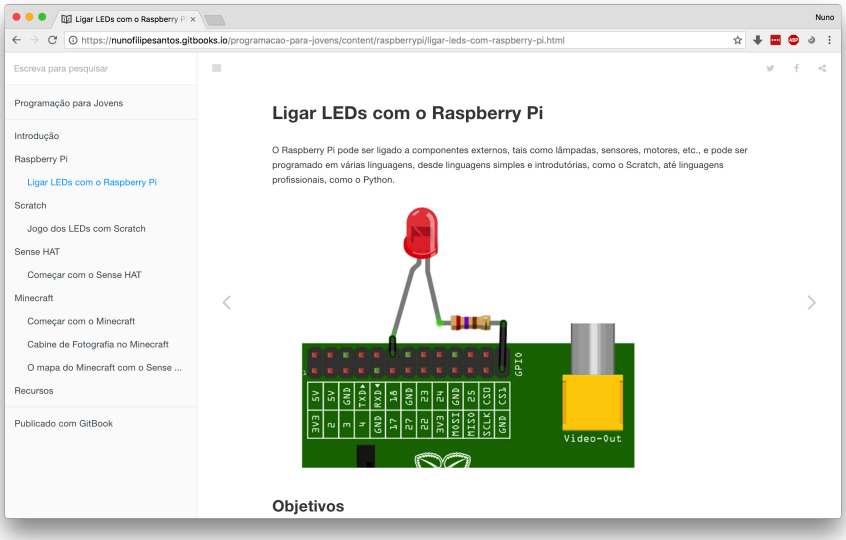


Figura B.4: Atividade: Ligar LEDs com o Raspberry Pi.

B.3 Scratch

Figura B.5: Capítulo: Scratch.

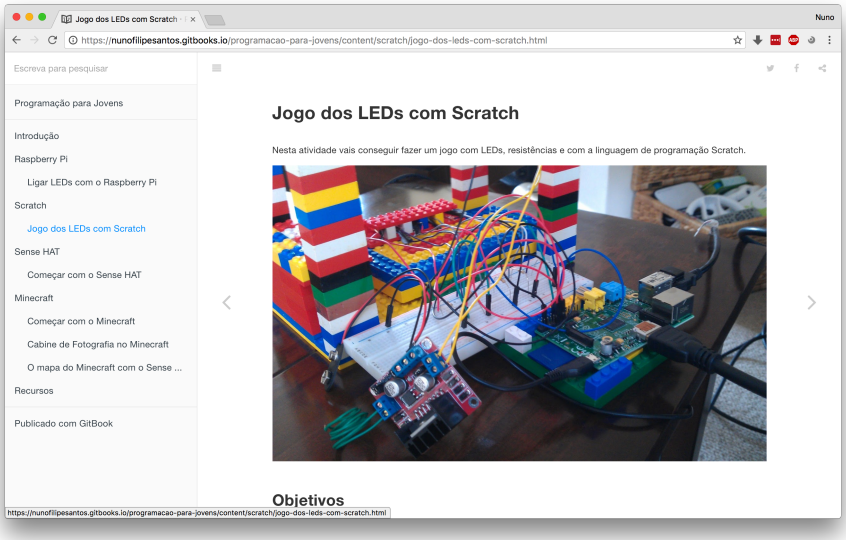


Figura B.6: Atividade: Jogo dos LEDs com Scratch.

B.4 Sense HAT

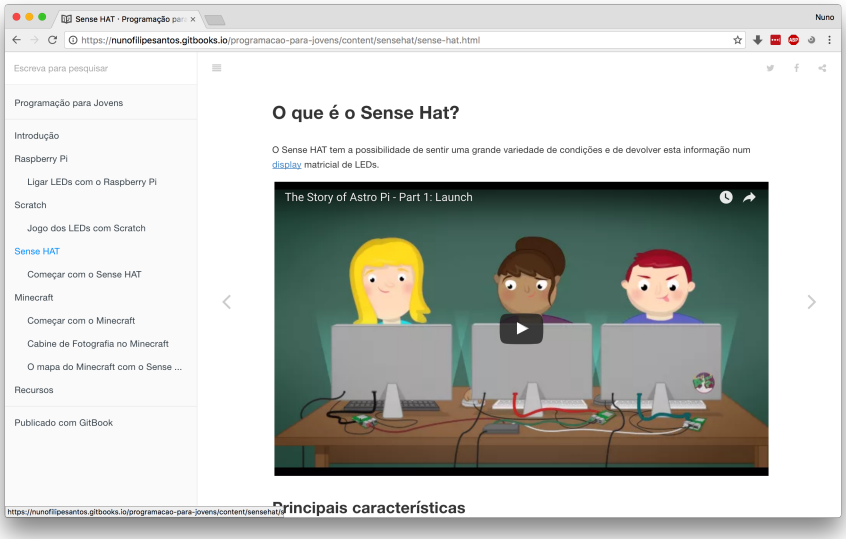


Figura B.7: Capítulo: Sense HAT.

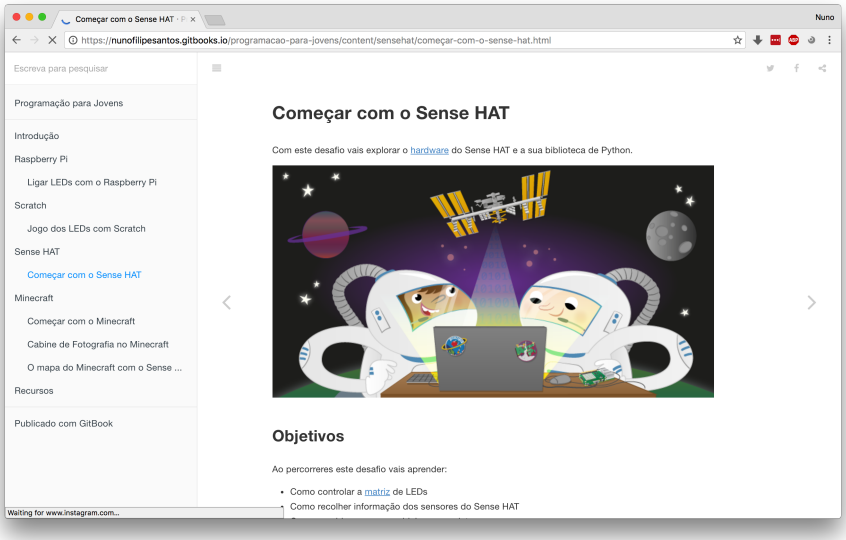


Figura B.8: Atividade: Começar com o Sense HAT.

B.5 Minecraft

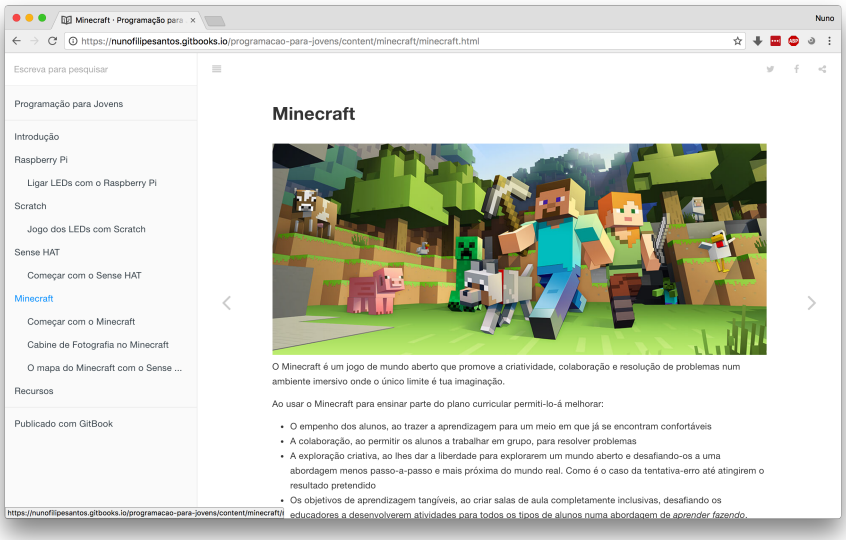


Figura B.9: Capítulo: Minecraft.

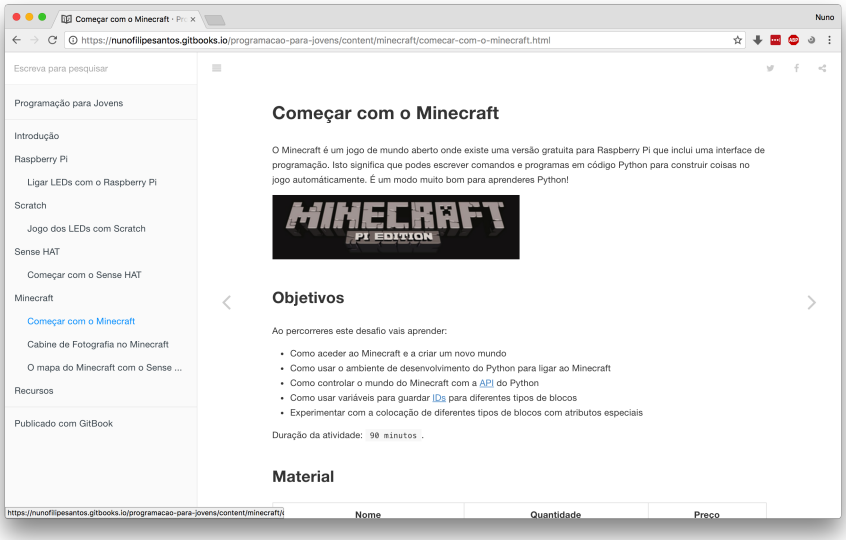


Figura B.10: Atividade: Começar com o Minecraft

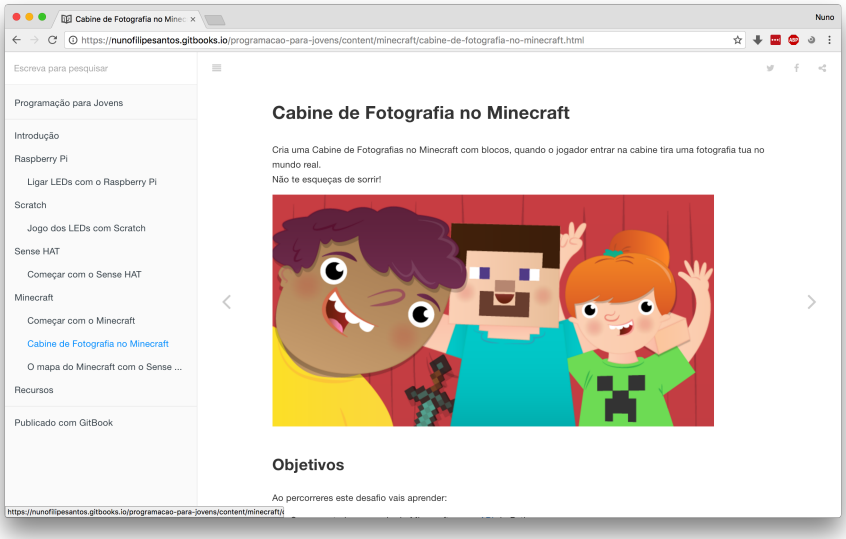


Figura B.11: Atividade: Cabine de Fotografia no Minecraft.

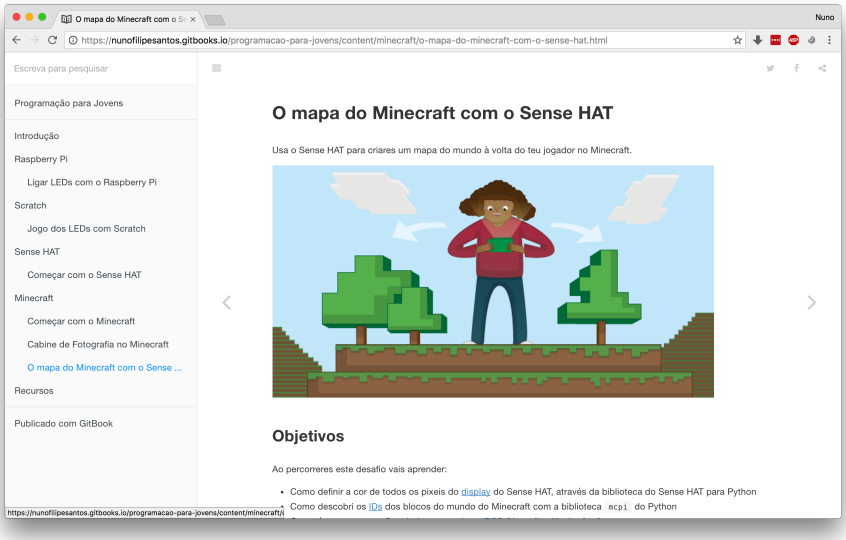


Figura B.12: Atividade: O mapa do Minecraft com o Sense HAT.

B.6 Recursos

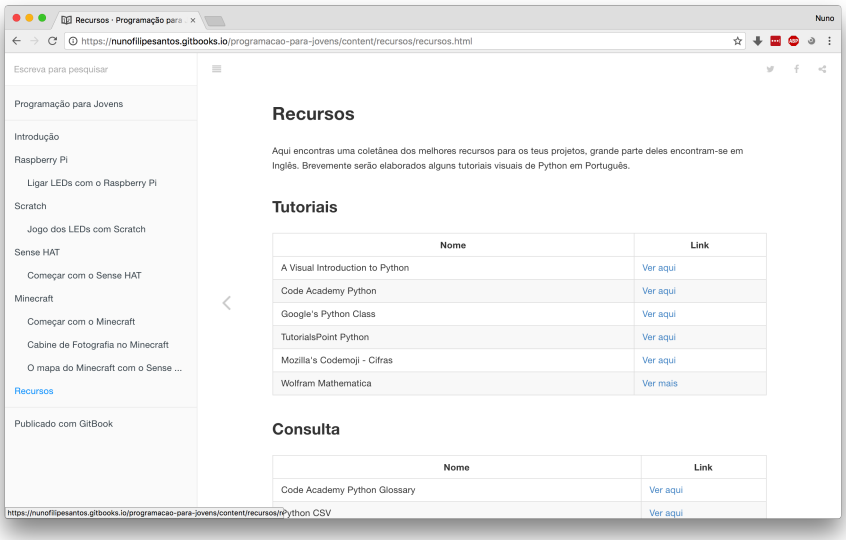


Figura B.13: Recursos do livro.

B.7 Glossário

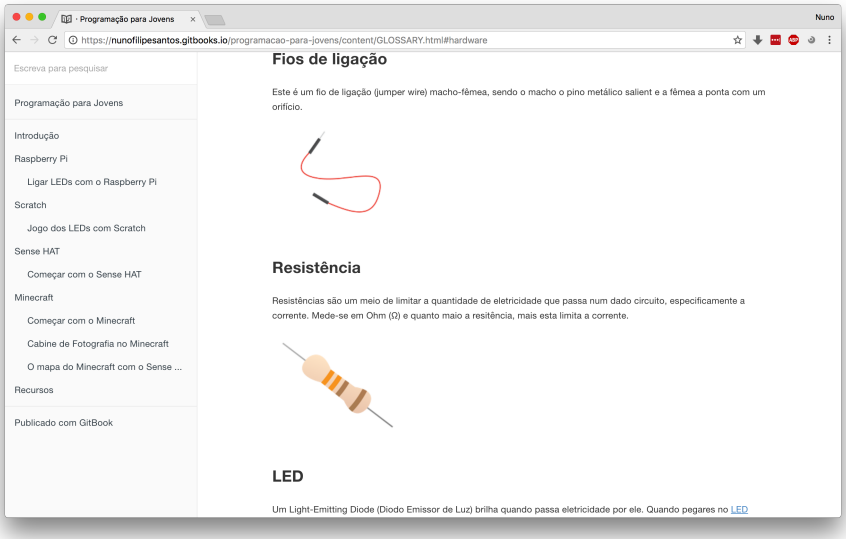


Figura B.14: Glossário do livro.

Referências

[20115] RASPBERRY PI GOES TO ISS FOR SCHOOLS CODING COMPETION Verify and Optimize your Designs. 44(January):451580, 2015.

[AK00] Ritu Agarwal e Elena Karahanna. Time Flies when You're Having Fun:Cognitive Absorption and Belief about Information Technology Usage. *MIS Quarterly*, 24(4):665–694, 2000.

[All] The Scrum Alliance. The Scrum framework in 30 seconds. page 30. URL: https://www.scrumalliance.org.

[ASO16] Ademar Aguiar, Nuno Santos e Outros. Programar é Fácil, 2016. URL: https://github.com/Coding4Kids/programarefacil/wiki.

[Ass97] Association for the Advancement of Computing in Education. Journal of Interactive Learning Research, 1997. URL: https://www.aace.org/pubs/jilr/.

[BCD+16] Stefania Bocconi, Augusto Chioccariello, Giuliana Dettori, Anusca Ferrari e Katja Engelhardt. *Developing Computational Thinking in Compulsory Education*. 2016. URL: http://publications.jrc.ec.europa.eu/repository/bitstream/JRC104188/jrc104188{_}computhinkreport.pdf, doi:10.2791/792158.

[CCD+15] Andrew Csizmadia, Paul Curzon, Mark Dorling, Simon Humphreys, Thomas Ng, Cynthia Selby e John Woollard. Computational thinking: A guide for teachers. 2015.

[Cha95] Daniel Chandler. Technological or media determinism, 1995. URL: http://www.waena.org/ktm/week1/TechorMediaDeterminism.pdf, doi:10.4135/9781446215159.n355.

[CST12] CSTA. Computer Science Teachers Association, 2012. URL: https://csta.acm.org.

[DBOM99] BENEDICT DU BOULAY, TIM O'SHEA e JOHN MONK. The black box inside the glass box: presenting computing concepts to novices. *International Journal of Human-Computer Studies*, 51(2):265–277, 1999. doi:10.1006/ijhc.1981.0309.

[EEG+05] Barbara Ericson, Barbara Ericson, Mark Guzdial, Mark Guzdial, Maureen Biggers e Maureen Biggers. A model for improving secondary CS education. *ACM SIGCSE Bulletin*, 37(1):332, 2005. URL: http://portal.acm.org/citation.cfm?doid=1047124.1047460, doi:10.1145/1047124.1047460.

[EGM14] Barbara J Ericson, Mark Guzdial e Tom Mcklin. Preparing Secondary Computer Science Teachers Through an Iterative Development Process. pages 116–119, 2014.

[Geo08] Fotis Georgatos. How applicable is Python as first computer language for teaching programming in a pre-university educational environment, from a teacher's point of view? (June):135, 2008. URL: http://arxiv.org/abs/0809.1437.

[Git17] Inc. GitHub. GitHub Education, 2017. URL: https://education.github.com.

[GP13] Shuchi Grover e Roy Pea. Computational Thinking in K12 : A Review of the State of the Field. *Educational Researcher*, 42(1):38–43, 2013. doi:10.3102/0013189X12463051.

[GPC14] Shuchi Grover, Roy Pea e Stephen Cooper. Promoting active learning & leveraging dashboards for curriculum assessment in an OpenEdX introductory CS course for middle school. *Proceedings of the first ACM conference on Learning @ scale conference - L@S '14*, pages 205–206, 2014. URL: http://www.scopus.com/inward/record.url?eid=2-s2.0-84899699274{&}partnerID=tZOtx3y1, doi:10.1145/2556325.2567883.

[JSS04] Andrew T. Jeffers, Angela G. Safferman e Steven I. Safferman. Understanding K–12 Engineering Outreach Programs. *Journal of Professional Issues in Engineering Education and Practice*, 130(2):95–108, 2004. doi:10.1061/(ASCE)1052-3928(2004)130:2(95).

[Kol13] Steve Kolowich. The Professors Who Make the MOOCs. *The Chronicle of Higher Education*, 2013. URL: http://chronicle.com/article/The-Professors-Behind-the-MOOC/137905/{#}id=overview.

[Lho07] Ladislav Lhotka. *History of UNIX*. 2007.

[LMD+11] Irene Lee, Fred Martin, Jill Denner, Bob Coulter, Walter Allan, Jeri Erickson, Joyce Malyn-Smith e Linda Werner. Computational thinking for youth in practice. *Acm Inroads*, 2(1):32–37, 2011. URL: http://dl.acm.org/ft{_}gateway.cfm?id=1929902{&}type=html{%}5Cnpapers3://publication/doi/10.1145/1929887.1929902, doi:10.1145/1929887.1929902.

[Min16] Minecraft-Mojang.AB. Minecraft: Education Edition, 2016. URL: http://education.minecraft.net/announce011916/.

[Moj17] Mojang. Minecraft, 2017. URL: https://minecraft.net/.

[MR16] Joshua C. Benson McCarthy e Brian R. SERVER-BASED AND SERVER-LESS BYOD SOLUTIONS TO SUPPORT ELECTRONIC LEARNING. 2016.

[PEH07] K Powers, S Ecott e L M Hirshfield. Through the looking glass: Teaching CS0 with Alice. *SIGCSE 2007: 38th SIGCSE Technical Symposium on Computer Science Education*, pages 213–217, 2007. doi:10.1145/1227310.1227386.

[Ric99] Jeffrey Richter. Programming applications for Microsoft Windows. 1, 1999.

[RN16] Karl Royle e Jasmina Nikolic. A modern mixture, Agency, Capability, Technology and 'Scrum': Agile Work Practices for Learning and Teaching in Schools. *Journal of Education & Social Policy*, 3(3), 2016.

[RSK+09] Mitchel Resnick, Brian Silverman, Yasmin Kafai, John Maloney, Andrés Monroy-Hernández, Natalie Rusk, Evelyn Eastmond, Karen Brennan, Amon Millner, Eric Rosenbaum e Jay Silver. Scratch. *Communications of the ACM*, 52(11):60, 2009. URL: http://dl.acm.org/ft{_}gateway.cfm?id=1592779{&}type=html, doi:10.1145/1592761.1592779.

[Scr11] ScrumAlliance. Scrum Alliance, 2011. URL: http://www.scrumalliance.org/pages/what{_}is{_}scrum.

[Tho00] John W. Thomas. A review of research on project-based learning. 200.

[THP93] Walter F. Tichy, Nico Habermann e Lutz Prechelt. Summary of the Dagstuhl workshop on future directions in software engineering, 1993. doi:10.1145/157397.157399.

[Tri15] Trinket.io. Trinket, 2015. URL: https://trinket.io/.

[WWY02] Laurie Williams, Eric Wiebe e Kai Yang. In support of pair programming in the introductory computer science course. *Computer Science Education*, 3(12):197–212, 2002. doi:10.1076/csed.12.3.197.8618.

[ZFS+15] Alexey Zagalsky, Joseph Feliciano, Margaret-Anne Storey, Yiyun Zhao e Weiliang Wang. The Emergence of GitHub as a Collaborative Platform for Education. *Proceedings of the 18th ACM Conference on Computer Supported Cooperative Work & Social Computing*, pages 1906–1917, 2015. doi:10.1145/2675133.2675284.

[ZWCLJ13] Christopher Zorn, Chadwick A Wingrave, Emiko Charbonneau e Joseph J LaViola Jr. Exploring Minecraft as a conduit for increasing interest in programming. *Fdg*, pages 352–359, 2013.